UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E ENERGIA

Desenvolvimento de um de Sistema de Monitorização Remota de Condições de Conforto Interior em Edifícios

Rúben José Silvina Teixeira

Dissertação

Mestrado Integrado em Engenharia da Energia e do Ambiente

2012

UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E ENERGIA

Desenvolvimento de um Sistema de Monitorização Remota de Condições de Conforto Interior em Edifícios

Rúben José Silvina Teixeira

Dissertação orientada pelos

Prof. Doutor Guilherme Carrilho da Graça (FCUL)

Prof. Doutor Jorge Maia Alves (FCUL)

Mestrado Integrado em Engenharia da Energia e do Ambiente

2012

Abstract

Indoor comfort, in buildings, has become increasingly relevant. In modern societies the majority of time is spent in indoor areas, and for that it is important that the occupants feel satisfied with the environmental conditions that surround them. Besides, this is a factor that heavily contributes to an increased energetic consumption in buildings, and it is also known to affect the occupants' productivity while performing their activities. For this reason it is important to monitor indoor conditions so as to evaluate and classify the indoor environment quality.

The purpose of this dissertation was to design, develop and test a monitoring system of indoor comfort conditions (ALVI). This system processes data in real-time, using wireless communication, locally storing information, and includes its own interface to visualize the data in a computer, while making the same data available online.

In this particular context the concept of open source, due to the availability of information and also due to the fact that is a free technology it allows the development and assembly of a device of this sort for a low price, appears as an alternative to the devices which are already available in the market, while having the same exact functions.

Keywords: buildings, indoor comfort, indoor environment quality, monitoring system

Resumo

O conforto interior nos edifícios tem vindo a ganhar cada vez maior relevância. Nas sociedades atuais a maioria do tempo é passado em espaços fechados, por isso é importante que os ocupantes se sintam satisfeitos com as condições ambientais que os rodeiam. Além disso, este é um fator que contribui fortemente para o elevado consumo energético nos edifícios e afeta também a produtividade dos ocupantes no desempenho das suas tarefas. Neste sentido é importante monitorizar as condições interiores de forma a poder avaliar e classificar a qualidade do ambiente interior.

O objetivo desta dissertação foi o de projetar, desenvolver e testar um sistema de monitorização de condições de conforto interior (ALVI). O sistema faz o processamento dos dados em tempo real, utiliza comunicação sem fios, armazena informação localmente, inclui uma interface própria para visualização num computador e disponibiliza dados online.

Neste contexto, o conceito *open source*, devido à disponibilidade de informação e tecnologia livre permite construir um equipamento deste tipo a baixo custo que pode ser uma alternativa aos dispositivos já existentes no mercado, apresentando as mesmas funcionalidades.

Palavras-chave: edifícios, conforto interior, qualidade do ambiente interior, sistema de monitorização

Índice

Lista de Figuras

Lista de Tabelas

Siglas Utilizadas

AC/DC	Alternating Current/Direct Current
ADC	Analog to Digital Converter
ALVI	Aluno Virtual
AVAC	Aquecimento Ventilação e Ar-Condicionado
COMS	Complementary metal-oxide-semicondutor
CSV	Comma-separated values
EEPROM	Eletrically-Erasable Programmable Read-Only Memory
GND	Ground (massa)
I2C	Inter-Integrated Circuit
IP	Internet Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
NDIR	Non-dispersive infrared
OEM	Original equipment manufacturer
PDE	Processing Development Environment
PPD	Predicted Percentage Dissatisfied
PMV	Predicted Mean Vote
PMW	Pulse-Width Modulation
ppm	Partes por milhão
QAI	Qualidade do Ar Interior
QCI	Qualidade do Clima Interior
RMS	Root mean square
RTC	Real time clock
SCL	Serial Clock Line
SD	Secure-Digital
SDA	Serial Data Line
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP-IP	Transmission Control Protocol, Internet Protocol
UART	Universal Asynchronous Receiver/Transmitter.
USB	Universal Serial Bus
Vcc	Tensão de alimentação em corrente contínua
VR	Valor Referência
WLAN	Wireless Local Area Network

1. Introdução

"During the last few years, due to the improvement of living standards, and the solutions provided by technological progress, the general expectation for high quality indoor environment has gradually increased. Because of the large percentage of time (often more than 90%) people spend indoors, the need to maintain indoor comfort conditions and at the same time to improve the energy efficiency is a crutial subject." (Corgnati, et al., 2011)

Atualmente, uma das principais finalidades dos edifícios é proporcionar um ambiente saudável e confortável para os seus utilizadores e para o desenvolvimento das suas atividades. Os projetos de construção, em comparação com o passado, têm de respeitar requisitos mais exigentes como a segurança, eficiência energética, sustentabilidade e qualidade do ambiente interior.

O conceito de conforto interior, nos edifícios, tem-se alterado ao longo do tempo, no que diz respeito aos aspetos que se relacionam com a satisfação dos ocupantes. Nas sociedades contemporâneas, em que grande parte do tempo é passado em espaços fechados, a qualidade do ambiente interior dos edifícios tem vindo a ganhar uma especial relevância. As condições de conforto interior são afetadas pelos equipamentos de aquecimento, ventilação e ar condicionado (AVAC), iluminação, arquitetura, materiais de construção, ocupantes, e ainda pela operação e manutenção do edifício (Bronsema, et al., 2004). Estas condições devem ser mantidas para que não influenciem de forma negativa o conforto dos ocupantes. Nos últimos anos, vários estudos comprovam a existência de uma forte influência da qualidade do ambiente interior no bem-estar e desempenho dos mesmos. Estes apresentam diferentes níveis de produtividade, na prática de atividades de escritório e educação, quando sentem sensação de conforto ou de desconforto (Corgnati, et al., 2011).

Numa altura em que cada vez mais a preocupação mundial é atingir níveis de eficiência elevados, nomeadamente na área dos edifícios, um dos grandes desafios consiste em reduzir o consumo energético dos mesmos, ao mesmo tempo que se conserva a qualidade do ambiente interior. Assim, para evitar problemas no clima interior, é essencial que a otimização energética seja integrada com a avaliação do mesmo. A eficiência energética e o conforto estão inteiramente ligados, o que significa que a aplicação de uma melhoria num destes aspetos só é aceitável se não afetar negativamente o outro. Por exemplo, tem que se ter em atenção que, por vezes, na tentativa de melhorar o desempenho energético existe o perigo de comprometer o bem-estar e a perceção de conforto dos utilizadores, e vice-versa.

Quando se fala das edificações já existentes, estas representam, neste momento, a maior área de intervenção, muito por culpa do paradigma económico que se vive nos dias de hoje na Europa. A construção de imóveis está um pouco estagnada e a tendência será para, nos próximos anos, remodelar e otimizar os edifícios já construídos. As instituições de ensino superior, num cômputo geral, estão inseridas nesse contexto.

Recentemente, na Universidade de Lisboa iniciou-se um projeto denominado "Universidade Verde", que tem como objetivo o desenvolvimento de um campus mais sustentável, tanto a nível energético como ambiental. Os edifícios de serviços, como é o caso das instalações do campus da Universidade de Lisboa, têm como seus principais consumidores os sistemas AVAC e de iluminação. Por conseguinte, será também importante fazer uma avaliação das condições de trabalho existentes nos espaços como salas de aula, laboratórios e gabinetes, que são pontos de possível intervenção e por vezes apresentam um clima interior inadequado para as tarefas aqui realizadas. No que diz respeito ao condicionamento do conforto interior, o campus apresenta atualmente uma situação bastante heterogénea, com uma variação entre espaços não climatizados e espaços com soluções de climatização local, normalmente feita por sistemas do tipo split (Alves, et al., 2011). Posto isto, torna-se importante fazer uma monitorização das condições do ambiente interior, com o intuito de perceber se este se encontra ou não na zona de conforto, seja ele térmico, visual, acústico ou qualidade do ar e, por outro lado, verificar a qualidade dos sistemas existentes.

O presente trabalho tem como principal finalidade desenvolver um sistema portátil que possibilite fazer a monitorização remota e avaliação das condições de conforto interior dos espaços no campus da Universidade de Lisboa. Após a construção do protótipo, será efetuado um período de testes, com recolha de dados, para aferir se este funciona em boas condições e recolhe dados fiáveis. Os dados recolhidos serão utilizados para fazer uma análise das condições de conforto existentes num determinado espaço.

Este sistema deve ser complementado com um equipamento de monitorização do consumo elétrico (objeto de desenvolvimento no âmbito de outra Dissertação a decorrer em paralelo) pois, como mencionado anteriormente, o condicionamento do conforto interior é um fator que influencia bastante os gastos energéticos. Assim, devido à sua forte relação, a análise destes dois aspetos deve ser feita em conjunto.

2. Qualidade do ambiente interior

O ser humano saudável sente sensação de conforto quando está satisfeito com as condições ambientais do meio envolvente. O conceito de conforto interior é bastante complexo e difícil de definir devido ao seu caráter subjetivo, uma vez que depende de fatores físicos, fisiológicos e psicológicos, ou seja, varia de pessoa para pessoa. No entanto, este pode ser caraterizado pela interpretação de estímulos, própria de cada indivíduo, manifestando-se através de reações fisiológicas e/ou emoções.

Num edifício, o nível de conforto associado ao bem-estar está relacionado com vários parâmetros, que podem ser de natureza pessoal (estado mental, vestuário, hábitos, sexo, idade, atividade, etc.) como ambiental (iluminação, nível de ruído, qualidade do ar, ambiente térmico, espaço disponível por ocupante, etc.) (Groth, 2007; Corgnati, et al., 2011; Khedari, et al., 2000).

A qualidade do ambiente interior ou qualidade do clima interior (QCI) pode ser descrita como um índice de medição que engloba o efeito combinado das condições de conforto térmico, conforto visual, conforto acústico e qualidade do ar interior na sensação de bem-estar dos ocupantes. Esta está dependente das condições exteriores (ruído, temperatura exterior, luz solar, etc.), dos sistemas de condicionamento do conforto interior, da operação e manutenção do edifício, bem como das suas características de construção (Alfano, et al., 2010).

Condições inadequadas do ambiente interior podem levar a problemas de saúde tanto a curto como a longo prazo, sendo que os primeiros são normalmente reversíveis e desaparecem assim que a causa ou fonte de desconforto é removida ou atenuada. Por outro lado, uma boa qualidade do ambiente interior aumenta o nível de conforto e contribui para um melhor desempenho e produtividade dos ocupantes. No caso específico de uma sala de aula, a uma QCI adequada ajuda a reduzir a distração, permitindo o aumento dos níveis de concentração e mantendo alunos e professores saudáveis (Corgnati, et al., 2011). Por outro lado, os sistemas associados à promoção da QCI contribuem decisivamente para o consumo energético nos edifícios. Também por esta razão o interesse na análise e avaliação das condições de conforto tem vindo a ganhar ainda mais ênfase.

2.1 Conforto térmico

O conforto térmico é descrito como uma condição da mente que expressa satisfação com o ambiente térmico (ASHRAE, 2004). Este depende de fatores ambientais como a temperatura do ar, humidade relativa, temperatura média radiante, velocidade do ar e assimetrias térmicas, de fatores individuais como a atividade metabólica[1] e a resistência térmica do vestuário[2] (ISO 7730) e ainda de variáveis fisiológicas como a temperatura da pele, transpiração, etc. A sua definição não é exata nem está afeta a valores de temperatura nem de humidade relativa definidos, pois depende da perceção individual de conforto. Na prática é impossível conseguir satisfazer todos os ocupantes de um mesmo espaço devido às diferenças psicológicas e fisiológicas de cada indivíduo.

A sensação de conforto térmico está diretamente relacionada com o balanço térmico das trocas de calor operadas entre o corpo humano e o ambiente que o rodeia. A geração de calor processa-se através da produção de calor pelo metabolismo e as perdas são feitas por condução, evaporação, radiação e convecção através da pele e da respiração. O corpo humano possui um mecanismo termorregulador que mantém a temperatura corporal mais ou menos constante a um valor médio de

[1] Expressa em *met* ou W/m^2, onde 1 *met* representa 58,1 W/m^2 de calor gerado por um indivíduo normal, que equivale a uma taxa metabólica bastante baixa, ou seja, um indivíduo sentado.

[2] Expressa em *clo*, que traduz o isolamento térmico do vestuário nas trocas de calor do corpo humano, onde 1 clo = 0.155 $m^2°C/W$. Quanto maior o valor de clo menores são as perdas de calor para o meio.

37ºC. Existe conforto quando o organismo perde exatamente o calor que necessita para não sentir frio nem calor.

A expressão do balanço de energia entre o corpo e o ambiente é descrita através da equação (1) (ASHRAE, 2009):

$$M - W = q_{sk} + q_{res} + S \quad (1)$$

Em que:

M = Taxa metabólica de produção de calor (W/m^2)
W = Trabalho mecânico realizado pelo corpo (W/m^2)
q_{sk} = Taxa total de perda de calor através da pele (W/m^2)
q_{res} = Taxa total de perda de calor através da respiração (W/m^2)
S = Taxa total de armazenamento de calor no corpo (W/m^2)

O modelo mais utilizado para caracterização das condições de conforto térmico foi desenvolvido por Fanger, sendo que serviu de base à elaboração da Norma Internacional ISO 7730. Este modelo estacionário assume que o corpo humano, quando exposto a um ambiente térmico constante, mantém a sua temperatura constante, sem acumulação de calor (S=0), uma vez que existe um equilíbrio entre a produção de calor e as suas perdas.

Ole Fanger sugeriu uma equação de conforto baseando-se em experiências realizadas com pessoas e fazendo uso do conceito de balanco de energia. A equação de conforto foi concebida para determinar todas as combinações dos fatores ambientais e individuais que propiciam uma sensação térmica neutra ao corpo humano, ou seja, existência de um ambiente térmico em que um indivíduo não sente calor nem sente frio. Devido às diferenças existentes de pessoa para pessoa acima referidas, esta equação prevê as condições térmicas ambientais para as quais maior parte dos indivíduos estarão satisfeitos (Corgnati, et al., 2011).

A partir desta metodologia, Fanger também procurou quantificar as sensações de calor e frio criando um índice de sensação térmica, denominado Predicted Mean Vote (PMV), que procura traduzir a votação média previsível em relação a um determinado ambiente, segundo a escala de sensação térmica ASHRAE (ver Tabela 1). O índice PMV prevê a sensação térmica em função do nível de atividade, vestuário e parâmetros de ambiente térmico. Este estudo foi baseado num cálculo estatístico sobre a votação individual, relativamente ao grau de satisfação, de um grupo de pessoas sujeitas às mesmas condições térmicas.

Tabela 1- Escala de sensação térmica ASHRAE

+3	Muito Quente
+2	Quente
+1	Levemente Quente
0	Neutro
+1	Levemente Frio
+2	Frio
+3	Muito Frio

Outro parâmetro, igualmente desenvolvido através do modelo de Fanger, chamado Predicted Percentage Dissatisfied (PPD), corresponde à previsão da percentagem de pessoas insatisfeitas com um determinado ambiente térmico e pode ser calculado através do valor de PMV (ISO 7730). O PPD é importante pois tem em conta a diferença que existe na perceção de conforto de cada pessoa, englobando a percentagem de indivíduos que apesar de poderem estar num ambiente considerado neutro (PMV=0) apresentam insatisfação com o mesmo, devido às suas particularidades psicológicas

e fisiológicas. Isto pode ser observado na Fig. 1, onde, para um valor de PMV neutro existem cerca de 5% de pessoas insatisfeitas. Segundo o índice de PPD, pessoas que votem -3,-2,+2 ou +3 na escala PMV são consideradas termicamente insatisfeitas. O PPD tem uma relação bastante direta com o PMV, pois quando o último referido se encontra num intervalo entre -0,5 e +0,5, recomendado pela ISO 7730, o PPD não apresenta valores superiores a 10%, como mostra a Fig. 1.

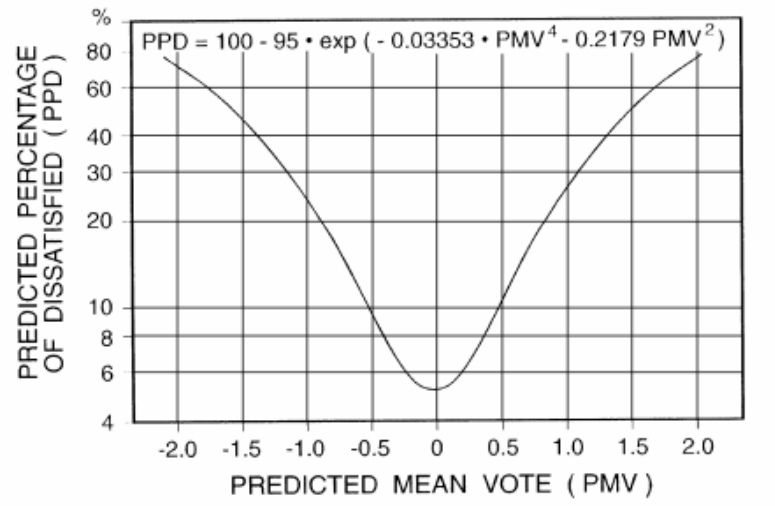

Fig. 1 - PPD em função do PMV (ASHRAE, 2004)

Estes parâmetros (PMV e PPD) estabelecem condições de satisfação ou insatisfação do ambiente de uma forma global. No entanto poderá haver ocasiões em que os ocupantes experimentam uma sensação de desconforto térmico local devido a um elevado gradiente de temperatura vertical, a uma temperatura do solo demasiado elevada ou reduzida, a uma grande assimetria da temperatura radiante ou ao efeito de circulação do ar. As pessoas com baixo nível de atividade física apresentam uma maior suscetibilidade a estas situações de desconforto local do que pessoas com elevado nível de atividade física (Corgnati, et al., 2011). No que concerne aos fatores que afetam as condições de desconforto local, pode também ser calculada a percentagem de ocupantes incomodados pelo efeito da circulação do ar (DR = Draught Rate) e a percentagem de ocupantes insatisfeitos (PD = Percentage Dissatisfied) no que diz respeito às assimetrias radiativas.

Temperatura operativa

Para um determinado espaço e em condições estabelecidas de humidade, velocidade do ar, taxa metabólica e isolamento do vestuário, pode ser calculado um intervalo de temperaturas operativas. Este intervalo corresponde a uma zona de conforto que fornece os limites de condições térmicas aceitáveis ou combinações de temperatura do ar e temperatura radiante consideradas termicamente aceitáveis (ASHRAE, 2004). A temperatura operativa caracteriza-se como a temperatura correspondente a condições de ambiente térmico neutro (PMV = 0). Esta pode ser calculada através de uma média ponderada entre a temperatura do ar e a temperatura média radiante.

De uma forma geral, pode fazer-se uma aproximação da temperatura operativa, considerando-a igual à temperatura do ar, nos casos em que não há diferenças de temperatura significativas entre as superfícies do espaço e a temperatura do ar, o espaço não possui qualquer painel radiante e os ganhos solares são reduzidos (ASHRAE, 2004). Para a realização deste projeto é razoável fazer esta aproximação, não só pelas características dos espaços a monitorizar, que vão ao encontro do referido acima, como também pelo erro associado a uma medição feita num local específico, que já introduz

por si só uma incerteza nos valores de temperatura global. Isto porque, para diferentes locais específicos, num espaço interior, as temperaturas apresentam uma certa variação (a temperatura do ar no espaço não é perfeitamente homogénea, podendo variar por exemplo em altura e com a proximidade de fontes de calor internas). Deste modo torna-se difícil conseguir medir a temperatura média global de todo o espaço interior, sendo esta uma limitação do sistema que é difícil ultrapassar.

Humidade relativa

A humidade relativa, correspondente à quantidade de vapor de água existente numa massa de ar em relação ao máximo que esta poderia conter à mesma temperatura, é outro dos parâmetros a ser analisado pelo sistema desenvolvido neste trabalho. Apesar de não ser um fator determinante no conforto térmico para níveis de atividade baixos, como é o caso de estudo presente, a exposição por longos períodos de tempo a níveis de humidade demasiado elevados ou demasiado reduzidos pode causar desconforto, problemas de saúde aos ocupantes ou reduzir a qualidade do ar, sendo por isso importante a sua monitorização. Um valor reduzido de humidade relativa (abaixo dos 30%) pode propiciar o aumento de desconforto e a secagem das membranas mucosas e pele, conduzindo à formação de gretas e irritações, enquanto níveis elevados (acima dos 70%) podem originar fenómenos de condensação nas superfícies interiores dos espaços e consequente desenvolvimento de fungos (Silvestre, 2011).

2.2 Qualidade do ar

A qualidade do ar interior (QAI) num determinado espaço relaciona-se com a presença de substâncias poluentes como gases, odores, vapores e partículas de origem biológica ou mineral que possam causar desconforto ou risco de prejudicar a saúde dos ocupantes. Os indivíduos expostos a uma qualidade do ar interior pobre ou inadequada podem manifestar sintomas a curto ou a médio prazo, como por exemplo irritação ou vermelhidão nos olhos, garganta seca/irritada, dores de cabeça, tonturas e fadiga. Também a produtividade e desempenho podem ser afetadas por este fator (Alfano, et al., 2010). Estas substâncias têm origem no exterior (indústria, automóveis, etc.) ou no interior do edifício, sendo produzidas e libertadas pelos ocupantes, pelos equipamentos do edifício ou pelos materiais de construção ou limpeza. Normalmente, para eliminar o excesso destas substâncias, procede-se à ventilação do espaço, através de abertura de janelas ou dos sistemas AVAC.

A avaliação da QAI pode ser efetuada pela análise da concentração, no ar interno, de um ou mais poluentes. Um dos principais poluentes encontrados num espaço interior é o CO_2, sendo que a sua concentração é usada, com frequência, como indicador da qualidade do ar interior dos edifícios. Este parâmetro é um bom indicador da QAI para situações em que a produção dos poluentes do ar está maioritariamente relacionada com os ocupantes. Para os casos em que esta não é a fonte maioritária, torna-se um critério de interesse limitado, visto que não fornece uma indicação completa e global da qualidade do ar.

O CO_2 é um gás incolor, inodoro e insípido, sendo a sua concentração numa amostra de ar geralmente expressa em partes por milhão (ppm). Atualmente a concentração de dióxido de carbono presente na atmosfera terrestre, em zonas pouco poluídas, é, em média, de 380 ppm (Alfano, et al., 2010). Dentro de um espaço interior, um ensaio experimental efetuado numa sala de aula sem renovação de ar e com 26 ocupantes (durante uma hora) teve como valor máximo medido 2645 ppm, sendo que os valores mínimos estão próximos de 500 ppm (Silvestre, 2011).

Num edifício de serviços, os ocupantes são os principais intervenientes na produção dos poluentes, nomeadamente de dióxido de carbono, expelido durante a respiração. A sua concentração é dependente do número de pessoas presentes, da correspondente fisiologia, intensidade de atividade física bem como da taxa de renovação de ar.

Alguns estudos experimentais comprovaram, de igual modo, a existência de uma relação direta entre a concentração de CO_2 no ar e o índice de aceitabilidade do espaço no que diz respeito aos odores causados pelos biofluentes humanos, também produzidos pelos ocupantes e relacionados com a atividade física, dieta e higiene. Estes ensaios fazem uma correlação entre a percentagem de pessoas insatisfeitas (recém-chegadas) com o odor dos biofluentes e a concentração de CO_2 acima do exterior. Esta relação é usada como indicador para aferir sobre a QAI num espaço, em que a única fonte de poluição são os ocupantes como se pode ver na Fig. 2 (Emmerich, et al., 2001; Prill, 2000).

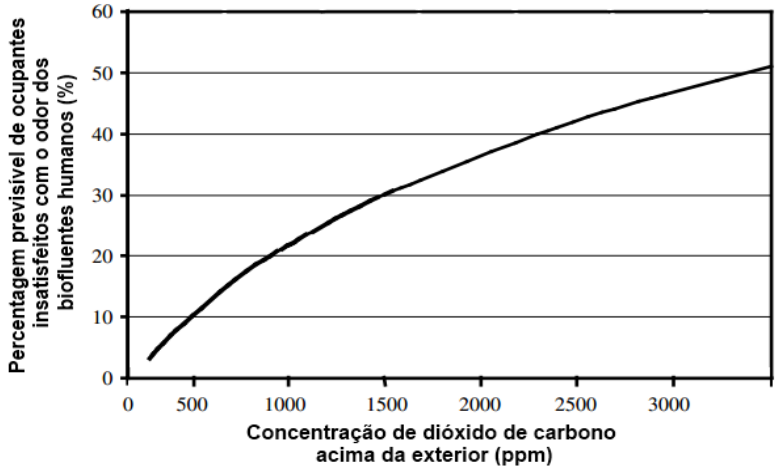

Fig. 2 - Percentagem previsível de ocupantes insatisfeitos com o odor dos biofluentes humanos em função da concentração de CO_2 acima da exterior (Emmerich, et al., 2001, modificado)

Segundo o gráfico da Fig. 2, de modo a manter, pelo menos, 80% dos ocupantes recém-chegados satisfeitos com o nível de odores causados pelos biofluentes humanos, deve assegura-se que a concentração de CO_2 no ar interior não ultrapassa 700 ppm em relação ao exterior.

2.3 Conforto visual

Este parâmetro de conforto é principalmente afetado pelas condições de iluminação existentes num determinado espaço e pelas tarefas desempenhadas dentro do mesmo. O ambiente de iluminação tem o objetivo de assegurar boas condições de visibilidade para que as tarefas visuais sejam executadas de forma confortável e depende do tipo e do período de utilização do espaço ou edifício. Se um determinado espaço apresentar um ambiente com iluminação inadequada, pode causar ou agravar problemas de visão ou provocar sintomas como fadiga visual, irritabilidade ocular, dores de cabeça, dores musculares e dificuldade de concentração (Pais, et al.).

No caso de um edifício educacional, onde as atividades de leitura e aprendizagem são constantes, o ambiente visual afeta as capacidades dos estudantes, assim como o seu desempenho. Além disso a luz influencia o relógio biológico e a regulação de certas hormonas (Alfano, et al., 2010). Por exemplo em condições de iluminação reduzida, é frequente o aparecimento de sonolência.

A norma europeia EN 12464-1 define as exigências de iluminação necessárias para proporcionar um bom ambiente de trabalho em espaços interiores.
Um bom ambiente visual, no que toca a requisitos de iluminação, deve respeitar as necessidades quantitativas e qualitativas para desempenhar uma determinada tarefa.
Estes requisitos de iluminação podem ser alcançados através da satisfação das seguintes exigências humanas básicas:

- Conforto visual - a sensação de bem-estar de um indivíduo reflete-se numa elevada produtividade.
- Desempenho visual - o indivíduo consegue executar bem as tarefas visuais, mesmo em circunstâncias difíceis e durante períodos mais longos.
- Segurança

O ambiente visual, segundo a norma EN 12464-1, deve ser caracterizado através de cinco parâmetros: a distribuição de luminosidade, a iluminância, o brilho, a reprodução de cor e aparência de cor da luz e a luz do dia (luz natural).

Dentro dos referidos, a iluminância será o fator de conforto visual monitorizado pelo sistema construído neste trabalho, visto ser o que afeta de forma mais direta o conforto visual. A iluminância (E_v), também denominada por nível de iluminação, é a quantidade de fluxo luminoso - quantidade de luz emitida por uma fonte de luz (lúmen) - incidente sobre uma superfície, por unidade de área. Diminui com o quadrado da distância, expressa-se em lux, e pode ser calculada através da seguinte equação (2) (Zumtobel Staff, 2004):

$$E_v = \frac{\Phi_v}{A} \tag{2}$$

Em que:

E_v =Iluminância (lux)

Φ_v = Fluxo luminoso (lm)

A = Área da superfície (m^2)

São normalmente utilizados valores de iluminância mantida para avaliar os espaços em relação a conforto visual. A iluminância mantida é o valor mínimo para o qual o nível de iluminância deve ser assegurado no plano de trabalho e também depende do tipo de tarefa a ser desempenhada neste local. Esta varia no espaço de acordo com a posição do plano de trabalho em relação à fonte de luz, portanto é necessário ter em atenção onde é feita a medição.

2.4 Conforto acústico

Uma boa qualidade de conforto acústico é essencial para manter os ocupantes satisfeitos, concentrados e com bom desempenho nas suas tarefas. Para este critério de conforto a variável de maior interesse é o ruído. Não existe nenhuma diferença física entre som e ruído, todavia este último pode definir-se como um som indesejável que provoca uma sensação auditiva desagradável, incómodo ou até problemas auditivos. A propagação de ruído num espaço depende bastante do nível de isolamento acústico da construção.

Uma onda sonora pode ser caracterizada pela sua frequência, intensidade e duração. Existem várias maneiras de caracterizar o ruído, normalmente numa escala logarítmica.

De um modo geral é usado o decibel[3], dB, para exprimir o nível de pressão sonora. No entanto, como o ouvido humano reage de forma diferente aos sons de diversas frequências e tem uma maior sensibilidade a frequências médias (ex. voz humana), utiliza-se um filtro de ponderação A, que traduz

[3] O decibel caracteriza-se pela razão logarítmica entre uma determinada pressão sonora e o valor referência.

aproximadamente a resposta do aparelho auditivo humano (Arezes, 2002). As medições efetuadas através de um filtro de ponderação de frequências A tem como unidades o dB(A).

A sensibilidade do ouvido humano adulto tem os seus limites dentro do intervalo de frequências entre 20 Hz e 20 000 Hz, e os limites de intensidade vão de 0 (limiar de audição) a 130 dB (limiar de dor) (PCB Piezotronics, Inc.). Nos espaços a ser estudados, não se prevê que exista um nível de ruído alto o suficiente que se torne perigoso para o aparelho auditivo. Contudo, no que diz respeito às salas de aula ou escritórios/gabinetes, um som indesejado afeta negativamente os níveis de concentração dos ocupantes, interfere na comunicação entre eles e, consequentemente torna o ambiente impróprio para realizar tarefas de escritório e/ou de aprendizagem. Neste tipo de espaços o conforto acústico pode ser influenciado por ruído proveniente de fontes exteriores como o tráfego rodoviário, ferroviário, aéreo, indústria, etc., ou de fontes interiores como conversação entre os ocupantes, circulação de pessoas ou ruído dos equipamentos instalados, por exemplo, sistema de AVAC. No contexto da aprendizagem, é muito importante a comunicação entre professores e alunos. A inteligibilidade do discurso, que se define pela capacidade de compreender palavras faladas, diminui com o aumento do ruído (Alfano, et al., 2010).

Neste projeto, devido ao material disponível para medição deste parâmetro, não será possível fazer medições diretas de dB(A). Deste modo, por limitações do sensor de ruído a utilizar, será feita uma calibração do mesmo por comparação com um sensor calibrado, que efetua medições em valores dB(A). Assim a avaliação do conforto acústico será mais no sentido de qualificar do que quantificar, derivado ao facto de a calibração do sensor ter alguns erros associados. Portanto os valores finais obtidos serão apenas valores aproximados de dB(A).

3. Avaliação da qualidade do ambiente interior

Após a apresentação dos conceitos teóricos sobre conforto interior em edifícios, neste capítulo será feita uma abordagem geral aos objetivos e aspetos a ter em conta na realização de uma campanha de monitorização deste género, tal como da tecnologia existente para esse fim. Além disso, serão também definidos os critérios de classificação do ambiente interior e definidos os parâmetros a medir, assim como os correspondentes valores limite regulamentados, com base nas principais normas internacionais e europeias que abordam esta temática.

3.1 Monitorização

A monitorização da qualidade do ambiente interior, num edifício, é uma tarefa que engloba diferentes variáveis, as quais têm que ser cuidadosamente ponderadas e planeadas, de modo a tornar o processo mais preciso e tentar minimizar os erros de medição. É necessário definir bem o objetivo da campanha de monitorização e tomar decisões ao nível do período de tempo pelo qual se vão estender as medições, da localização dos sensores, dos parâmetros a medir e da frequência das medições.

Uma campanha de monitorização deste tipo pode ser direcionada para dois principais objetivos, nomeadamente, a classificação da qualidade do clima interior ou diagnóstico da mesma (Corgnati, et al., 2011).

A classificação permite:

- atribuir categorias de QCI através de índices específicos;
- comparar o nível atual de QCI com as exigências dos ocupantes;
- comparar e verificar as variações da QCI entre diferentes espaços e/ou edifícios;
- comparar os valores medidos com os valores recomendados na regulamentação.

O diagnóstico tem como finalidade:

- perceber a evolução da QCI ao longo do tempo;
- destacar as causas da falhas e queixas num edifício;
- propor possíveis soluções para melhorar o nível de QCI ou poupar energia.

Os parâmetros a medir dependem do que se pretende e do nível de detalhe que se define para a campanha de monitorização. Idealmente, para uma análise completa, deveriam ser medidos e estudados todos os fatores que afetam o conforto, no entanto isto nem sempre é possível.

O período de medição pode ser longo ou curto, variando de alguns dias ou semanas até um ano inteiro, dependendo do que se pretende. Uma medição curta permite fazer uma análise a períodos críticos do ano, por exemplo os dias de extremos de temperatura, ou locais críticos do espaço, como zonas de desconforto local. Por seu turno, uma medição longa é feita para espaços em que é necessário analisar as variações temporais dos parâmetros de clima interior.

A localização mais adequada dos sensores, segundo Corgnati, et al., (2011), deve ser numa zona central do espaço ou na região onde se encontram os ocupantes, mas sempre a mais de um metro de distância das paredes.

3.1.1 Sistemas de monitorização existentes

Neste momento existem disponíveis no mercado diversas opções com a capacidade de medir os parâmetros de conforto e que podem ser utilizadas na monitorização da qualidade do ambiente interior. De um modo geral, um sistema deste tipo tem as seguintes características:

- Sensores para medição dos parâmetros
- Memória para armazenar os dados medidos
- Ecrã para visualização em tempo real
- Portabilidade

A maioria destes dispositivos faz a medição de apenas um ou dois parâmetros isoladamente. No entanto, começam a ser cada vez mais frequentes aqueles que monitorizam vários parâmetros ao mesmo tempo.

Como exemplo destes sistemas portáteis podem encontrar-se o IAQ-Calc 7545 desenvolvido pela empresa TSI e o MI-6201 Multinorm produzido pela empresa Metrel. (ver Fig. 3).

Fig. 3 – MI-6201 Multinorm à esquerda e IAQ-Calc 7545 à direita (fonte: Metrel e TSI)

O IAQ-Calc é utilizado na avaliação da qualidade do ar, medindo em simultâneo a concentração de CO_2, CO, e ainda dados de conforto térmico, neste caso a temperatura do ar e a humidade relativa. No que diz respeito ao MI-6201 Multinorm, este é capaz de medir a temperatura do ar, a humidade relativa, a iluminância, o nível de ruído e a velocidade do ar, possibilitando ainda a inclusão de outros acessórios de medição e funcionalidades disponibilizadas pela marca. A medição dos dados dos parâmetros acima referidos é feita apenas para 3 sensores em simultâneo, por limitação do número de portas de ligação. Ambos os aparelhos mencionados têm visor integrado, são alimentados a pilhas e funcionam como *dataloggers* armazenando os dados medidos na memória interna. Paralelamente, disponibilizam um software para descarregar a informação no computador.

Existe ainda o HAZ-SCANNER IMS (Fig. 4), denominado pelo fabricante (SCK Inc.) como uma estação para monitorização da qualidade ambiente interior. Trata-se de um dos aparelhos mais completos a ser comercializado para este tipo de aplicações, pois permite medir até 14 parâmetros em simultâneo. Na sua configuração padrão inclui 5 sensores (partículas em suspensão, CO_2, CO, temperatura e humidade relativa) mas é suscetível à ligação de sensores adicionais para medição de intensidade luminosa, gases tóxicos, compostos orgânicos voláteis (VOC's), som, entre outros, num total de 9 opções. O aparelho tem um visor, usa uma bateria com autonomia para 8 horas, guarda os registos dos dados medidos e possibilita transmissão da informação por wireless em tempo real. Inclui ainda um software que permite gerar gráficos e relatórios no computador.

Fig. 4 - Estação HAZ-SCANNER IMS à esquerda na configuração padrão e à direita com alguns sensores adicionais

3.2 Classificação

Como referido anteriormente, devido a fatores que dizem respeito ao conforto dos ocupantes e ao tempo que se passa hoje em dia dentro de edifícios, quer a trabalhar, quer na casa de habitação, a monitorização e classificação do ambiente interno são aspetos importantes a ter em atenção. Também o interesse da certificação energética dos edifícios é tido em conta, como não era no passado, e é agora associado ao condicionamento do clima interior.

Neste sentido, o Organismo Europeu de Normalização (CEN) elaborou a norma de conforto europeia EN 15251, " Parâmetros ambientais interiores para a conceção e avaliação do desempenho energético dos edifícios abordando a qualidade do ar interior, ambiente térmico, iluminação e acústica", baseada nas normas ASHRAE 55 e ISSO 7730, já existentes.
Segundo a norma Europeia EN 15251, a informação sobre o ambiente interior deve constar no certificado de desempenho energético de um edifício, de modo a que a sua performance total possa ser avaliada. O principal objetivo desta norma é especificar critérios de conforto para o ambiente interior, definindo parâmetros de entrada para o projeto de edifícios, previsão de consumos energéticos, performance energética e operação dos mesmos.

Com o intuito de tornar a avaliação da qualidade do ambiente interior mais simples, prática e fácil de compreender, nesta norma, foram estabelecidas diferentes classes ou categorias de classificação. Estas podem ser escolhidas para o espaço a ser condicionado, e a sua seleção depende do tipo de edifício, tipo de ocupantes e tipo de disparidades climáticas e nacionais. São ainda definidos os parâmetros a ser mostrados e monitorizados neste processo de classificação, tanto na fase de projeto como na fase de operação. Neste seguimento, são considerados no documento os agentes influenciadores da qualidade do ambiente interior: conforto térmico, qualidade do ar, conforto visual e conforto acústico.

Tabela 2 - Âmbito de aplicação das categorias de classificação dos edifícios no que diz respeito ao nível de expectativa (EN 15251, 2007)

Categoria	Explicação
I	Alto nível de expetativa e é recomendada para espaços ocupados por pessoas muito sensíveis e frágeis com necessidades especiais como deficientes, doentes, crianças muito pequenas e idosos
II	Nível normal de expetativa e deve ser utilizado para novas construções e renovações
III	Nível aceitável e moderado de expetativa e pode ser utilizado para os edifícios existentes
IV	Valores fora dos critérios para as categorias anteriores. Esta categoria só deverá ser aceite para uma parte limitada do ano

Como se detalha na Tabela 2, foram indicadas as categorias de classificação, que estão de acordo com os diferentes graus de aceitação do ambiente interior, estabelecidas pela previsão da percentagem de

ocupantes satisfeitos. Deste modo, e com a finalidade de definir e comparar diferentes níveis de qualidade do ambiente interior, requerida ou verificada, foram fixadas quatro categorias que correspondem a quatro escalões de expectativa: I (expetativa elevada), II (expetativa normal), III (expetativa moderada), IV (valores aceitáveis apenas para uma parte do ano).

Cada uma destas categorias especifica para os parâmetros de qualidade do ambiente interior referidos, quais os valores ou intervalo de valores recomendados, tendo em conta o nível de exigência considerado. Dentro das opções apresentadas, a categoria IV é a menos usada pois esta não impõe requisitos. Apenas se classifica um determinado espaço com a categoria IV quando os valores estão fora dos limites recomendados para as restantes categorias.

O sistema desenvolvido neste trabalho será capaz de monitorizar e avaliar, em tempo real, as condições de ambiente interior de acordo com os parâmetros e categorias referidos. O tipo de espaços que este estará preparado para avaliar são os seguintes:

- escritórios;
- cafés/restaurantes
- salas de aula;

Portanto, neste documento, serão apenas estudados e apresentados os valores recomendados para os aspetos referidos, bem como para os parâmetros de conforto a ser monitorizados pelo sistema aqui construído, também mencionados anteriormente. É de referir que para o caso presente foram escolhidos os dados e informação, constantes da norma, referentes aos critérios de projeto.

3.2.1 Valores referência para cada parâmetro

Os valores presentes nas tabelas que se seguem são os dados a utilizar pelo sistema a desenvolver para avaliação da QCI. Assim estes valores recomendados são utilizados para comparar com os valores medidos em tempo real pelo sistema.

- *Temperatura*

No que diz respeito ao ambiente térmico, a norma EN 15251 indica os valores de temperatura operativa que devem ser respeitados dentro do edifício (para o Verão e para o Inverno). Como referido anteriormente a temperatura operativa será, para todos os efeitos, considerada igual à temperatura do ar. Na Tabela 3 podem ser observados estes valores, para níveis de atividade e de isolamento de vestuário estabelecidos.

Tabela 3 – Valores de dimensionamento recomendados de temperatura operativa para escritórios, cafés/restaurantes e salas de aula considerando uma taxa metabólica baixa (1,2 met) e um isolamento de vestuário estabelecido para as estações de verão (0,5 clo) e inverno (1 clo)

Tipo de Espaço	**Categoria**	**Temperatura Operativa (ºC)**	
		Mínima (inverno)	**Máxima (verão)**
Escritórios, Cafés/Restaurantes e Salas de Aulas	I	21,0	25,5*
	II	20,0	26,0
	III	19,0	27,0
* Para salas de aula este valor é de 25ºC (na norma)			

Para efeitos de avaliação da QCI, apesar do valor definido na norma ser 25ºC, neste trabalho considerou-se, por aproximação, que a temperatura operativa máxima nas salas de aula é de 25,5ºC. Isto porque os valores referência para os escritórios e cafés/restaurantes são de 25,5ºC e uma diferença de 0,5ºC não tem muita relevância, podendo assim fazer-se esta aproximação de modo a ter somente um conjunto de condições de temperatura.

- *Humidade relativa*

Geralmente, a humidificação do ar interior não é necessária. Como referido anteriormente, esta tem um efeito relativamente reduzido sobre a sensação térmica e perceção da qualidade do ar em locais com ocupação sedentária. Todavia, na norma, também são definidos valores padrão para este parâmetro, que podem ser observados na Tabela 4. Estes são valores estabelecidos para dimensionar sistemas de humidificação e desumidificação para espaços ocupados. Os locais a ser monitorizados no desenvolvimento deste trabalho não dispõem deste tipo de sistemas, contudo, estes dados serão utilizados no caso de estudo como valores referência para avaliação da humidade relativa.

Tabela 4 - Valores limite recomendados para a humidade relativa num espaço ocupado segundo a norma EN15251

Tipo de Espaço	**Categoria**	**Humidade relativa (%) mínima para humidificação**	**Humidade relativa (%) máxima para desumidificação**
Espaços onde os critérios de humidade são estabelecidos pelos ocupantes	I	30,0	50,0
	II	25,0	60,0
	III	20,0	70,0

Além disso, recomenda-se que a humidade absoluta deve ser mantida em valores abaixo de 12 g/kg.

- *Concentração de CO2*

Este é o único parâmetro da qualidade do ar interior aqui estudado. Os limites referenciados são estabelecidos para concentrações de CO_2 acima dos valores presentes no ar exterior. A norma recomenda, assim, os valores presentes na Tabela 5.

Tabela 5 - Valores máximos recomendados para a concentração de CO2 (ppm) acima da concentração no ar exterior (EN 15251)

Categoria	**Concentração de CO_2 (ppm) acima dos valores do ar exterior**
I	350
II	500
III	800

Por conseguinte, calculam-se os valores limite a ser utilizados para o presente trabalho, que resultam da soma dos valores médios no exterior (380 ppm) com os definidos na norma. Assim, obtiveram-se como concentrações limite a considerar: 730 ppm (categoria I), 880 ppm (categoria II) e 1180 (categoria III).

- *Iluminância*

No que toca ao conforto visual a norma não estabelece categorias, sendo que os valores a ser utilizados são iguais para as todas as três consideradas. Os valores referência para a iluminância mantida são assim estabelecidos pela Tabela 6.

Tabela 6 - Valores recomendados de iluminância mantida presentes na norma EN15251

Tipo de Espaço	**Iluminância mantida (lux) no local de trabalho**
Escritórios	500
Salas de aula para adultos	500
Cafés/Restaurantes	-

Segundo Zumtobel Staff, (2004), para manter o conforto visual, estes valores não devem ser inferiores a 67% dos valores recomendados. Este critério é utilizado para avaliação do ambiente visual neste trabalho. Na norma não são considerados valores máximos de lux que afetem o conforto visual.

- *Nível de ruído*

Em relação aos valores recomendados para o nível de ruído são definidos os valores máximos apresentados na Tabela 7. Para este critério de conforto não são especificados valores de acordo com as categorias mencionadas.

Tabela 7 - Valores máximos recomendados de nível de pressão sonora para os três tipos de espaços abordados (EN 15251)

Tipo de Espaço	**Nível de pressão sonora [dB(A)]**
Escritórios	35
Salas de aula para adultos	35
Cafés/Restaurantes	45

4. Sistema de monitorização proposto

4.1 Caracterização e arquitetura

O sistema de monitorização remota da qualidade do ambiente interior desenvolvido no presente trabalho vai ao encontro da filosofia de *open source* em toda a sua conceção, tanto a nível de hardware como a nível de software. Isto para se poder usufruir da tecnologia aberta, não só pela disponibilidade de informação existente como também por questões monetárias, pois permite que o projeto seja feito a baixo custo.

Uma vez que uma das principais finalidades do sistema é a monitorização de salas de aula, funcionando este como se fosse um aluno que avalia as condições de conforto existentes, decidiu-se dar-lhe o nome de "Aluno Virtual" (ALVI).

O ALVI é caracterizado por cinco objetivos principais:

- Capacidade de monitorização de parâmetros de conforto térmico, visual, acústico e qualidade do ar em tempo real
- Portabilidade
- Capacidade de armazenamento de dados
- Transmissão de dados sem fios (remoto)
- Interface gráfica no computador
- Disponibilização de dados na internet

A monitorização dos parâmetros de conforto tem como objetivo fazer uma avaliação do estado atual dos espaços do campus na Universidade de Lisboa, no que diz respeito às condições de ambiente interior existentes e que visam proporcionar conforto aos ocupantes no desempenho das atividades de ensino ou trabalho de escritório. Pretende-se, assim, analisar e identificar espaços que não correspondam às exigências requeridas e influenciem, de alguma forma, a saúde ou o rendimento dos ocupantes, na realização das suas tarefas. Esta monitorização é feita em sincronismo com o tempo real devido à utilização de um relógio de tempo real, que permite também o registo da hora e data em que é feita a recolha de dados.
Como o intuito é monitorizar todos os critérios de conforto, será analisada pelo menos uma variável para cada critério, de modo a ter uma amostra relativamente representativa. Em resumo, os parâmetros a ser medidos pelo sistema de monitorização desenvolvido neste projeto são os seguintes:

- Conforto Térmico – Temperatura e Humidade relativa
- Qualidade do Ar – Concentração de CO_2
- Conforto Visual – Iluminância
- Conforto Acústico – Nível de ruído

O facto de ser remoto e portátil faz deste sistema uma ferramenta bastante flexível, pois tem capacidade de comunicação à distância e sem fios, podendo, além disso, ser movimentado para qualquer local, sem problemas, devido ao seu tamanho e peso reduzidos. O armazenamento de dados no local é mais uma garantia de segurança da informação e para além disso permite o transporte da mesma para posterior acesso. Esta característica pode ser bastante vantajosa no caso de haver uma falha da comunicação sem fios, evitando deste modo a perda dos dados recolhidos.

A interface no computador disponibiliza uma avaliação imediata da qualidade do clima interior (por categorias), bem como gráficos da evolução dos parâmetros monitorizados, permitindo ao utilizador ter acesso ao estado atual do ambiente do espaço em estudo.

A utilização de um local de suporte de dados na internet faz com que a informação recolhida pelo ALVI possa ser acedida através do computador ou qualquer dispositivo com acesso à internet, estando disponível a qualquer hora e em qualquer lugar, sem as limitações associadas ao modo de acesso local.

Este protótipo divide-se de uma forma geral em duas partes distintas: uma parte de hardware e uma parte de software, sem as quais não poderia funcionar corretamente.

O hardware consiste em todos os elementos físicos que o compõem, nomeadamente a placa Arduino Uno, que é o constituinte mais importante, os sensores e relógio de tempo real para deteção e recolha de dados, e outros componentes que dão funções adicionais à placa, como a Shield Wireless SD com cartão micro-SD (micro Secure-Digital) e o módulo Wifly. Além disso, é ainda necessário um computador e um router, utilizados para possibilitar a visualização da interface criada e estabelecer a rede sem fios.

No que diz respeito ao software, utilizam-se programas que são igualmente parte integrante da plataforma *open source*, neste caso o Arduino IDE e o Processing, em que o primeiro é usado para desenvolver o software que o Arduino corre e para o programar e o segundo é uma ferramenta para programar interfaces gráficas.

Para que toda a eletrónica funcione, existem protocolos de comunicação que têm de ser aplicados, permitindo a transmissão da informação entre os sensores e o Arduino Uno. É também usada a tecnologia Wi-Fi, que permite uma forma de comunicação e transferência de dados à distância e sem necessidade de recorrer a uma infraestrutura ligada com fios.

De modo a possibilitar todas as funcionalidades implementadas pelo sistema desenvolvido, o fluxo de informação tem de passar por várias etapas até chegar ao utilizador. Os sensores são os responsáveis por fazer a conversão de uma grandeza física para sinais elétricos, que podem ser lidos e interpretados por microcontroladores. Estes sinais são posteriormente enviados, através das ligações existentes, para os pinos de entrada do Arduino Uno, onde podem ser lidos no microcontrolador aí existente. Por sua vez, a placa Arduino, se lhe forem adicionados os componentes necessários, consegue enviar a informação por uma rede sem fios, e por outro lado guardar os dados localmente num cartão micro-SD. Um router, configurado para estabelecer a rede sem fios, recebe a informação que é enviada remotamente e transmite-a para o computador, também via *wireless*. Quando chega ao computador, a informação pode ser visualizada na interface criada para o efeito, e é disponibilizada num servidor da web, o que permite o seu acesso em qualquer local que tenha acesso à internet.

O sistema deve ser alimentado a 12V, com um conversor AC/DC ou por USB. Este é suscetível de ser usado com uma fonte de alimentação própria (ex. bateria de 12V), sem necessidade de ligar ao sistema de eletricidade local, no entanto, não foi possível por motivos de prazo, ensaiar o tempo de autonomia nestas condições.

A descrição de todos os elementos que constituem o ALVI, bem como os seus princípios de funcionamento e as formas de comunicação, é feita de forma mais exaustiva e detalhada nos pontos que se seguem deste documento. Os diferentes processos e fases pelas quais a informação passa, serão abordados, também, com maior profundidade mais à frente.

A arquitetura e a forma como se processa o fluxo de dados, desde a deteção por parte dos sensores até ao utilizador, pode ser observada no esquema presente na Fig. 5, sendo que as setas indicam o sentido do fluxo de informação no sistema.

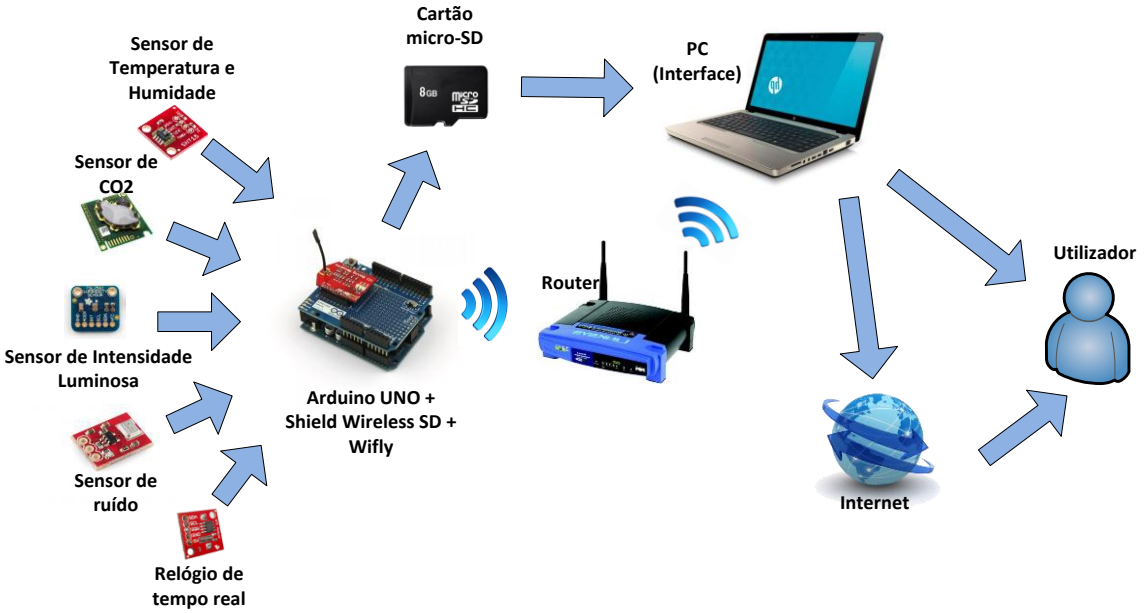

Fig. 5 - Diagrama representativo da arquitetura do sistema desenvolvido (ALVI) e direção do fluxo de informação

4.2 Hardware

4.2.1 Arduino

O Arduino é uma plataforma de desenvolvimento *open source* de hardware/software. É mais conhecido pela sua parte hardware, constituída por uma placa que contém um microcontrolador e todos os elementos eletrónicos necessários ao seu suporte, portas analógicas e digitais de entrada/saída, assim como pinos de alimentação. A criação desta placa foi inspirada num projeto chamado Wiring e com o intuito de ser a sua extensão física e eletrónica (Reas, et al., 2009). A sua parte de software permite programar o microcontrolador usando a linguagem de programação Arduino (baseada no Wiring) e o ambiente de desenvolvimento Arduino IDE (baseado no Processing). A placa de prototipagem é bastante flexível pois inclui possibilidade de comunicação com outros dispositivos através de interface serie USB, I^2C ou SPI. O microcontrolador (componente principal) pode ser programado a partir de um computador, via conexão USB (disponível nos pinos digitais 0 RX e 1 TX), sendo ele o responsável por controlar as entradas e saídas, armazenar o código e executá-lo (Banzi, 2011). A alimentação da placa Arduino é feita via USB, adaptador AC/DC ou com uma bateria, sendo a fonte selecionada automaticamente.

Todo este conjunto de ferramentas permitem fazer a interligação do mundo físico com o mundo virtual, ou seja, possibilita-nos perceber e responder a eventos físicos através de sensores e visualizar os dados por eles recolhidos.

O facto de apresentar um preço apelativo, um conceito "livre" e ter uma enorme gama de aplicações, faz dele o instrumento ideal para a elaboração do trabalho presente. O Arduino desempenha uma função nuclear na conceção deste projeto pois é através dele e dos seus instrumentos que se faz a interação com os sensores, recolhendo dados, armazenando-os e enviando-os via wireless para posteriormente serem tratados.

Arduino Uno

Existem várias versões desta placa, com diferentes componentes, tamanhos, objetivos etc., mas para este trabalho foi escolhida a mais recente no mercado e habitualmente mais utilizada, o Arduino Uno, que possui as funcionalidades necessárias para o desenvolvimento do mesmo, sendo recomendada como a mais simples de usar e a melhor para aprender (Banzi, 2011). Na Fig. 6 é apresentada uma imagem da placa Arduino Uno, incluindo a legenda dos seus principais componentes.

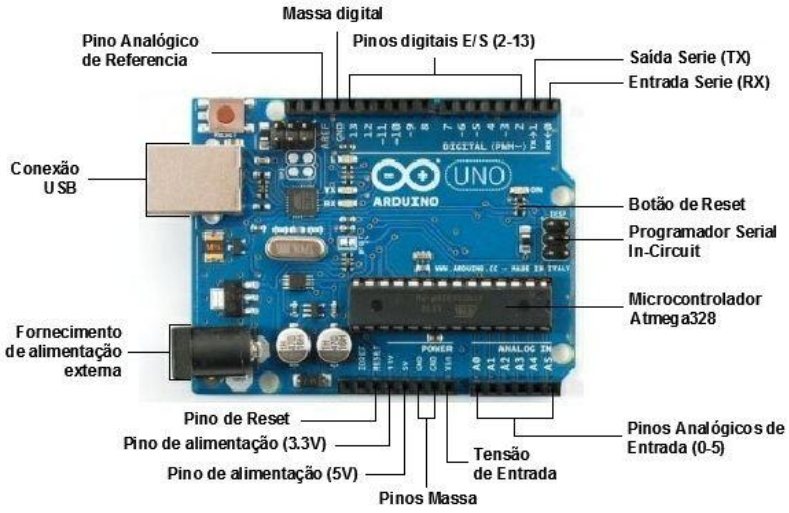

Fig. 6- Arduino Uno (fonte: http://www.watterott.com, modificada)

Principais Características (Arduino Team, 2012):

- Microcontrolador: Atmega328
- Tensão operacional: 5 V
- Tensão de entrada (recomendada): 7-12 V
- Tensão de entrada (limites): 6-20 V
- Pinos digitais de entrada/saída: 14 (dos quais 6 podem ser saídas PMW)
- Pinos de entrada analógica: 6 (10 bits)
- Corrente CC por pino entrada/saída: 40 mA
- Corrente CC para o pino 3,3V: 50 mA
- Memória Flash: 32 KB (Atmega328)
- SRAM: 2 KB (Atmega328)
- EEPROM: 1 KB (ATmega328)
- Interface de hardware: I2C, SPI, UART (5V)
- Velocidade do relógio: 16 MHz
- Dimensões: 75x53x15 mm

4.2.2 Wireless SD Shield

Uma das principais características do Arduino é a sua versatilidade e flexibilidade que, como referido anteriormente, nos possibilita a sua utilização numa enorme quantidade de aplicações. No entanto, este tem algumas limitações se usado de forma isolada, por isso neste conceito de *open source* é possível criar placas auxiliares, denominadas *shields* que podem ser a si acopladas e que lhe atribuem novas capacidades e funcionalidades.

No sistema desenvolvido para a presente dissertação, pretende-se que este tenha a capacidade de comunicar via *wireless* e ao mesmo tempo que armazene os dados recolhidos pelos sensores. Em concordância com estes requisitos, a Wireless SD Shield, foi escolhida por estar equipada de forma a poder proporcionar estas capacidades ao Arduíno Uno, com o qual é compatível.

Esta *shield* permite ao Arduino comunicar através de uma rede sem fios, Wi-Fi, se a esta for agregado um módulo de rádio wireless, como a Wifly (descrita mais à frente neste documento). Esta placa adicional foi desenhada para suportar qualquer módulo com formato idêntico ao rádio Xbee (tecnologia Zigbee), desenvolvido pela Digi, o que facilita, se assim pretendido, a transição de protocolo de comunicação apenas pela substituição do módulo de rádio. Para além disso, este componente auxiliar tem incluída uma área destinada a prototipagem e um encaixe para suporte de cartões micro-SD, que podem funcionar como memória amovível para armazenar informação proveniente, por exemplo de sensores, ou para ler dados guardados durante as medições. A Wireless SD Shield, na Fig. 7, tem os mesmos pinos de entrada/saída analógicos e digitais que o Arduino Uno, e quando encaixada estes assumem as funções dos pinos existentes no mesmo, com igual correspondência, excetuando o pino CS que também pode ser utilizado para comunicar com o cartão micro-SD.

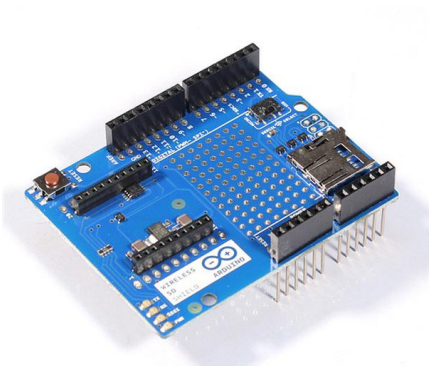

Fig. 7 - Wireless SD Shield (fonte: http://www.inmotion.pt)

Neste módulo auxiliar existe um interruptor, o *serial select*, que tem duas opções: Micro e USB. Quando está na posição Micro comunica com o microcontrolador, e a placa está configurada no modo *wireless* para transferência de dados ou para comunicar com o cartão micro-SD, não podendo o microcontrolador ser programado via USB. Na posição USB o módulo comunica diretamente com o computador para ser possível programar o microcontrolador (Arduino Team, 2012; SparkFun Electronics, 2012).

Para ter acesso ao cartão mico-SD utiliza-se uma biblioteca do Arduino, denominada “SD library”, sendo necessário que o pino CS da *shield* esteja livre e que o botão *serial select* esteja na posição Micro. A comunicação com o cartão é feita nesta entrada CS e nos pinos 11,12 e 13 onde é estabelecida uma ligação SPI.

4.2.3 Módulo Wifly

O módulo WiFly RN-XV (na Fig. 8), confere ao Arduino a capacidade de comunicar através de *wireless* pelo protocolo de comunicação Wi-Fi. Construído com base no módulo RN-171 da Roving Networks, segundo a norma IEEE802.11, e testado pela Wi-Fi Aliance, este dispositivo está equipado com rádio 802.11 b/g, processador SPARC de 32 bits, módulo TCP/IP, relógio, crypto acelerador, unidade de gestão de consumo e interface para sensores analógicos (Networks, 2011).

O seu design foi concebido à semelhança do rádio Xbee 802.15.4, ou seja, tem a mesma configuração física, para poder ser implementado de igual forma em hardware, como por exemplo a Shield

Wireless SD. Este módulo RN-XV já tem incluído *firmware* para tornar a sua integração mais rápida e fácil.

Para configurar a Wifly é necessário utilizar o software Teraterm fornecido pela Roving Networks, que funciona como um terminal onde é possível comunicar com o módulo de rádio, por ligação USB, transmitindo-lhe informação através de comandos específicos existentes no seu manual de comandos. Estes permitem configurar o módulo para se ligar a uma rede wireless, entre outras instruções. O Teraterm também pode ser utilizado como terminal para receber informação enviada remotamente pela Wifly.

A sua ligação com o Arduino/Wireless SD Shield é feita, por UART, através dos pinos RX e TX. É necessário utilizar uma biblioteca, a "Wifly-HQ" (https://github.com/) que facilita a comunicação do hardware pois tem funções e comandos específicos integrados.

Foi necessário ultrapassar alguns contratempos na integração do módulo Wifly com o Arduino Uno, pois a comunicação entre estes dois dispositivos é complexa e difícil de estabelecer. Existe pouca informação e exemplos sobre o modelo de hardware *wireless* escolhido por isso foi necessário analisá-lo pormenorizadamente, de modo a compreender qual a biblioteca do Arduino IDE a utilizar. Acresce que existe mais que uma biblioteca e cada uma é específica para um determinado modelo da Wifly, podendo uma escolha errada da mesma inviabilizar o correto funcionamento do hardware. Além disso, a configuração da Wifly, para permitir a ligação à rede sem fios, também não é simples, o que exigiu um estudo e aprendizagem sobre este tema.

Fig. 8 - Módulo WiFly RN-XV (fonte: http://www.sparkfun.com)

Principais características (Networks, 2011):

- Alimentação: 3,3 V
- Ligações: Vcc, TX, RX e GND
- Consumo: 4uA modo de suspensão, 38mA activo
- Interface de hardware: UART e SPI
- Taxa de recolha de dados superior a 464Kbps através de UART
- Suporta redes Adhoc e de infraestrutura (infrastructure networking)
- Autenticação de segurança Wifi suportada: WEP, WPA-PSK (TKIP), WPA2-PSK[4]
- Entradas/saídas digitais: 8
- Entradas analógicas para sensores: 3
- Memoria Flash: 8 Mbit
- Memoria RAM: 128 KB
- Relógio de tempo real para marcar o tempo, hibernar e ativar automaticamente
- Taxa de transmissão de dados: 1 – 11Mbps para 802.11b / 6 e 54Mbps para 802.11g
- Frequência: 2,4 GHz
- Dimensões: 24,4 x 34,3 mm

[4] O módulo Wifly RN-XV não suporta ainda o modo de segurança utilizado na *eduroam* (rede wireless da Universidade de Lisboa), o WPA2-Enterprise.

4.2.4 Sensor de temperatura e humidade

O sensor escolhido para fazer a medição dos parâmetros de temperatura e humidade relativa foi o SHT15 (Fig. 9), produzido pela Sensirion (Sensirion, 2011). É um módulo com chip, que tem em si acoplados um sensor de temperatura e um sensor de humidade relativa, devidamente calibrados e com sinal de saída digital. Para medir a temperatura utiliza um sensor de temperatura *band gap* e a humidade relativa é determinada por um sensor de polímero capacitivo. Ambos estão ligados a um amplificador, um conversor analógico/digital de 14 bits, memória OTP (On-time Programable) e um circuito de interface série, tudo integrado no mesmo componente. Isto resulta numa altíssima qualidade de sinal, grande precisão, baixo tempo de resposta e boa resistência (ou insensibilidade) a interferências externas.

Este dispositivo compacto tem aplicada a tecnologia COMS^{TM} (utilizada para construir circuitos integrados), conferindo-lhe esta característica grande fiabilidade e estabilidade.

A sua comunicação é feita por um protocolo semelhante ao I2C, adaptado pela Sensirion para otimizar a leitura e eficiência do sensor em termos de consumo, podendo assim ser ligado a qualquer microcontrolador. Uma vez que o protocolo não é exatamente igual, ainda que seja possível, não convém conectar este sensor a um barramento I2C, pois exige uma configuração bastante mais complexa do microcontrolador e pode mesmo interferir com os outros dispositivos ligados ao barramento.

Fig. 9 - Sensor de Temperatura e Humidade (SHT15) (fonte: http://www.robocore.net)

Principais caraterísticas:

- Alimentação: 2,4 a 5,5 V
- Ligações: Vcc, GND, SCK e DATA
- Interface de hardware: Digital 2-wire
- Intervalo de medição da humidade relativa: 0 – 100%
- Precisão da humidade relativa: ± 2%
- Intervalo de medição de temperatura: - 40ºC a +123,8ºC
- Precisão da temperatura: ± 0,3ºC
- Tempo de resposta: 8s
- Consumo médio: 90µW
- Dimensões: 20 x 19,5 mm

Este sensor, para ser ativado e efetuar as medições, necessita de receber uma sequência de comandos, provenientes do microcontrolador, que obedecem a uma ordem temporal. Os comandos de configuração são uma série de 8bits, enviados sob a forma de uma *string*, que o sensor consegue interpretar. Estes podem ser de iniciação, de medição, de verificação ou de *reset*. Os mais relevantes, neste contexto, são o comando para medição da temperatura "B00000011" e o comando para medição da humidade "B00000101".

Após as medições os dados lidos pelo sensor são enviados para o Arduino, em binário (SO_{RH} – humidade relativa e SO_T – temperatura), onde são feitos os cálculos de conversão para os valores reais

de temperatura e humidade relativa, utilizando o algoritmo apropriado, como descrito na ficha do produto disponibilizada pelo fabricante.

O valor de humidade relativa linear, sem compensação de temperatura, é calculado usando coeficientes de conversão próprios do sensor (c_1 = -2,0468; c_2 = 0,0367; c_3 = -1,5955E-6), através da equação (3):

$$HR_{linear}(\%) = c_1 + c_2 \times SO_{RH} + c_3 \times SO_{RH}^2 \quad (3)$$

O valor binário SO_T é convertido para o valor real de temperatura, pela equação (4):

$$T(^{o}C) = -40{,}1 + 0{,}01 \times SO_T \quad (4)$$

Sensor de temperatura *bandgap*

De uma forma geral, o princípio de funcionamento baseia-se na influência que a temperatura produz na tensão do díodo de silício. O sensor de temperatura presente no SHT15 é baseado numa célula de Brokaw, que consiste num circuito eletrónico utilizado para estabelecer a relação entre a tensão base-emissor (ΔV_{BE}) e a corrente (Ic) de dois transístores de junção bipolar (*p*-*n*)[5], cuja variação de corrente ou tensão é proporcional à temperatura absoluta.

Este fenómeno pode ser explicado através da equação (5) (Jung, 2005):

$$\Delta V_{BE} \cong V_{BE2} - V_{BE1} \cong \frac{kT}{q} \ln\left(\frac{I_{C2}}{I_{C1}}\right) \quad (5)$$

Onde:

V_{BE1} = Tensão base-emissor no transístor 1 (V)
V_{BE2} = Tensão base-emissor no transístor 2 (V)
I_{C1} = Corrente de coletor do transístor 1 (A)
I_{C2} = Corrente de coletor do transístor 2 (A)
k = Contante de Boltzmann ($1{,}38 \times 10^{-23}$ J/K)
q = Carga elétrica de um eletrão ($1{,}6 \times 10^{-19}$ C)
T = Temperatura absoluta (K)

Sensor de Humidade de polímero capacitivo

Este tipo de sensor consiste num condensador, cuja sua capacitância (C) varia de acordo com a humidade presente no ar, que neste caso é o material dielétrico. Ou seja, a maior ou menor quantidade de partículas de água na atmosfera influenciam a constante dielétrica (k) do ar, da qual depende a capacidade do condensador em armazenar carga elétrica. As equações (6) e (7) mostram que a

[5] Estrutura fundamental dos semicondutores, formada por dois metais do tipo P e N

constante dielétrica do ar húmido, e consequentemente a capacitância variam proporcionalmente com a humidade relativa (Fraden, 2010):

$$k = 1 + \frac{211}{T}\left(P + \frac{48P_S}{T}H\right)\cdot 10^{-6} \quad (6)$$

Em que:

k = Fator de permissividade elétrica do material dielétrico (adimensional)
T = Temperatura (K)
P = Pressão da mistura de ar (mmHg)
P_S = Pressão de saturação da água à temperature T (mmHg)
H = Humidade relativa (%)

$$C = k \cdot \varepsilon_0 \cdot G \quad (7)$$

Sendo que:

C = Capacitância do condensador (F)
ε_0 = Constante dielétrica (8,854 × 10^{-12} F/m)
G = Fator geométrico do condensador (m)

4.2.5 Sensor de CO_2

O Senseair K30, escolhido como sensor de CO_2 a usar neste projeto, é uma plataforma de medição de CO_2 que pode ser usada num vasto número de aplicações nas áreas da monitorização e do controlo. Foi construído para ser um módulo OEM (original equipment manufacturer) de modo a ser passível de integrar noutros aparelhos. É um sensor de alta qualidade e performance, individualmente calibrado, bastante estável e que não necessita de manutenção (CO2Meter.com, 2010).

O seu princípio de funcionamento assenta na tecnologia *non-dispersive infrared* (NDIR), que é bastante utilizada em estudos para análise do ar interior em edifícios. Este é um método que consiste, basicamente, em fazer passar um feixe de luz por uma célula de amostra (onde se encontra a amostra do gás que se quer saber a concentração), e, seguidamente, é medida no detetor a energia do feixe que atravessa a câmara. Pode observar-se o esquema de funcionamento deste tipo de sensores na Fig. 10.

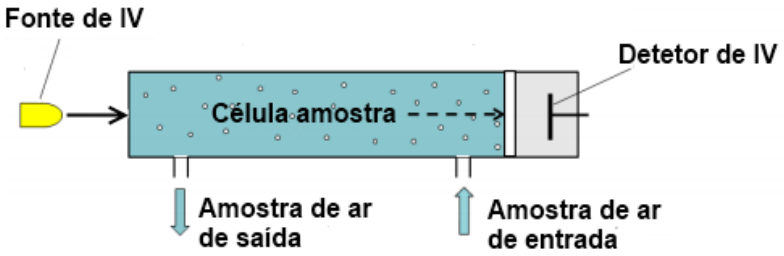

Fig. 10 - Diagrama representativo do funcionamento do sensor NDIR (baseado numa imagem retirada do Guidebook Indoor Climate Assessment)

O sensor funciona como um dispositivo espectroscópio, medindo a quantidade de radiação infravermelha, de um feixe de luz, que é absorvida pela amostra de ar. Cada gás absorve um comprimento de onda específico na região do infravermelho, sendo que o dióxido de carbono tem uma banda de absorção característica centrada no comprimento de onda de 4,26 µm (TSI Incorporated, 2009).

A quantidade de luz nesse comprimento de onda, que atinge o detetor, é inversamente proporcional à concentração de CO_2 presente na amostra, sendo assim possível determinar a sua concentração, neste caso, existente no interior do espaço onde é feita a medição. A relação entre a radiação que chega ao detetor e a concentração de CO_2 pode ser expressa pela equação (8), que respeita a equação de Lambert-Beer (Senseair, 2000):

$$I = I_0 \cdot e^{-cds} \tag{8}$$

Onde:

I = Intensidade de luz medida pelo detetor (W/m²)
I_0 = Intensidade de luz que incide na amostra (W/m²)
d = distância percorrida pela radiação (cm)
s = coeficiente de absorção do CO_2 ($dm^3\ mol^{-1}\ cm^{-1}$)
c = concentração de CO_2 (mol dm^{-3})

Para fazer a medição, o ar entra por difusão na célula de amostra, que está bem protegida por um filtro de partículas e foi especialmente desenhada para possibilitar uma rápida difusão, sem necessidade de insuflação forçada do gás.

O sensor K30, na Fig. 11, possui uma interface digital I2C, que permite ligação a qualquer microcontrolador. O seu sinal de saída reporta inclusivamente valores de concentração em unidades ppm. Este dispositivo está otimizado para controlo e monitorização de ambiente interior, apresentando medições bastante precisas.

Fig. 11 - Sensor de CO2 (Senseair K30) (fonte: http://www.co2meter.com)

Principais características:

- Alimentação: 4,5 a 9 V
- Ligações: G+, G0, pino A4 (SDA) e A5 (SCL)
- Interface de hardware: I2C
- Amplitude de medição: 0 a 5000 ppm
- Precisão: ±30 ppm + 5% do valor medido
- Sensibilidade: ±20 ppm + 1% do valor medido
- Método de amostra: difusão
- Temperatura de operação: 0 a 50 ºC
- Humidade de operação: 0 a 95% (sem condensação)
- Consumo médio: 40 mA
- Dimensões: 51 × 57 × 14 mm

4.2.6 Sensor de intensidade luminosa

O TSL2561, sensor de intensidade luminosa produzido pela Taos, converte a radiação que incide sobre si para um sinal digital, através de dois fotodíodos de banda larga. Capaz de medir dentro do espectro do infravermelho e do visível (contém um fotodíodo para Visível + IV e outro para IV), o sensor é dotado de dois conversores analógico-digital (ADC) que transformam, separadamente, a corrente dos díodos em sinais digitais de 16 bits, podendo estes ser reportados para qualquer microprocessador através de comunicação serie, I2C (Taos Inc., 2005).

O fotodíodo é um dispositivo eletrónico, construído com um material semicondutor de duas camadas, em que a junção *p-n* está exposta à luz. A camada *n* tem excesso de cargas negativas e a camada *p* tem excesso de cargas positivas. A junção *p-n* tem como propriedade o facto de funcionar como um conversor fotoelétrico. Este fenómeno acontece pois, quando a luz atinge o semicondutor, os eletrões existentes na estrutura são excitados e libertam-se dos átomos, deixando buracos (cargas positivas) no seu lugar. Forma-se assim o que se chama pares eletrão-buraco, sendo que o número de pares criados é uma medida do número de fotões de energia suficiente que penetram na estrutura. Na zona de depleção (região neutra), o campo elétrico estático, acelera os eletrões livres em direção à camada *n* e os buracos em direção à camada *p*. Isto cria polaridades na estrutura cristalina resultando numa corrente, que é proporcional ao fluxo de luz incidente. Na Fig. 12 pode observar-se um esquema do processo desencadeado quando a luz penetra na junção do semicondutor (Hamamatsu, 2010).

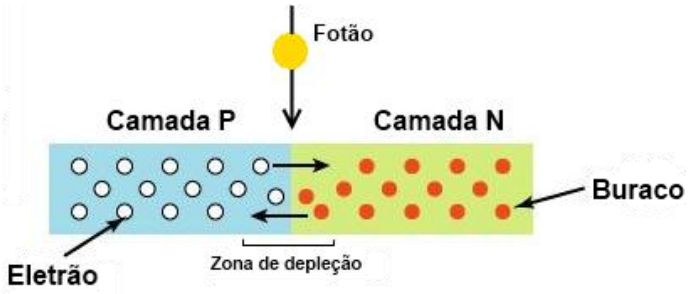

Fig. 12 - Esquema representativo do processo que ocorre quando um fotão penetra na junção *p-n* (fonte: http://electrapk.com/diode-biasing/, modificada)

Este sensor permite fazer determinações de iluminância (lux), e pode ser configurado para diferentes valores de ganho ou integração temporal de modo a detetar numa vasta gama de situações de iluminação. O facto de detetar nas bandas de infravermelho e "infravermelho + visível" de forma independente torna-o mais preciso em relação a outros sensores deste tipo, que normalmente utilizam apenas uma delas e por isso têm menos rigor na simulação da resposta do olho humano. Isto faz uma diferença significativa, pois algumas fontes de iluminação interior emitem uma grande percentagem de radiação no espectro do infravermelho. A resposta espetral para o canal 0 (visível + IV) e canal 1 (IV) do sensor pode ser observada na Fig. 13.

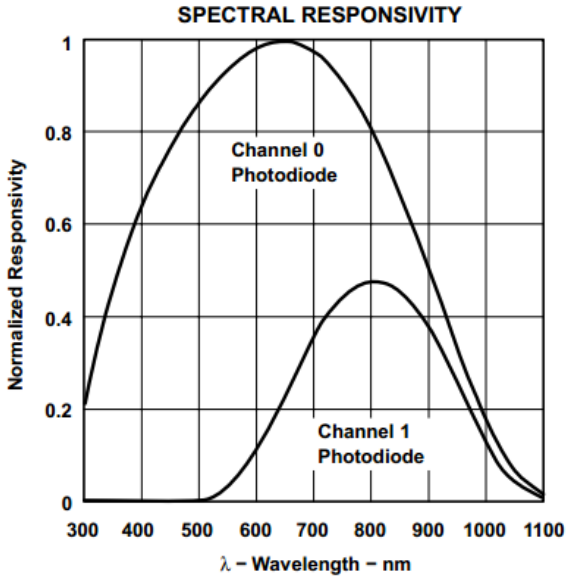

Fig. 13 – Resposta espectral dos fotodíodos de IV + Visível (canal 0) e apenas IV (canal 1) (fonte: ficha do produto)

O TSL2561 faz medições muito precisas e é bastante eficiente em termos energéticos, com uma corrente máxima de 0,6 mA quando está em funcionamento ativo, o que faz dele uma boa solução para integrar em sistemas embebidos de baixo consumo.

A integração deste sensor com o Arduino Uno é feita com uma biblioteca própria (TSL2561-Arduino-Library) que está disponível em *open source* (https://github.com) e que contém também toda a metodologia necessária para tratar os dados de saída digitais do canal 0 (CH0) e 1 (CH1) provenientes do conversor ADC. Estes dados são usados para obter um resultado aproximado à resposta do olho humano, que se pode ser expressa em valores de iluminância (Lux).

Os sinais de saída digital dos dois canais são analisados fazendo o quociente entre eles (CH1/CH0) e, dependo deste resultado, as equações usadas para o cálculo da iluminância são as presentes na Tabela 8.

Tabela 8 – Tabela de equações a utilizar para conversão dos valores provenientes do sensor em unidades Lux dependendo do rácio entre o espetro total e o espectro visível (fonte: ficha do produto)

CH1/CH0	**Equação a utilizar**
0 < CH1/CH0 < 0,52	Lux = 0,0315*CH0 – 0,0593*CH0 * ((CH1/CH0)^{1,4})
0,52 < CH1/CH0 ≤ 0,65	Lux = 0,0229*CH0 – 0,0291*CH1
0,65 < CH1/CH0 ≤ 0,80	Lux = 0,0157*CH0 – 0,0180*CH1
0,80 < CH1/CH0 ≤ 1,3	Lux = 0,0375 * CH0 – 0,0624 * CH1
CH1/CH0 >1,3	Lux = 0

As equações mostradas anteriormente foram obtidas tendo por base testes óticos realizados com fontes de luz incandescentes e fluorescentes.

Na Fig. 14 pode observar-se o sensor TSL2561, escolhido para medição de iluminância neste trabalho.

Fig. 14 - Sensor de intensidade luminosa (TSL2561) (fonte: http://www.ladyada.net)

Principais características:

- Alimentação: 2,7 a 3,6 V
- Ligações: Vcc, GND, SDA e SCL
- Interface de hardware: I2C e SMBus
- Aproximação à resposta do olho humano
- Amplitude de medição: 0,1 a 4000 lux
- Temperatura de operação: -30 a 70 ^{o}C
- Consumo médio: 3,2 µA (modo de suspensão), 0,24 mA (ativo)
- Dimensões 15 × 17,5 mm

4.2.7 Sensor de ruído

Para sensor de ruído foi escolhida a placa de circuito impresso para o microfone Electret, um microfone do tipo capacitivo. A sua estrutura consiste num diafragma móvel e uma placa de metal fixa que em conjunto formam um condensador de placas paralelas. Entre as placas existe um espaço de ar que funciona como material dielétrico (Fraden, 2010). Na Fig. 15 pode ser visto o esquema de construção do sensor.

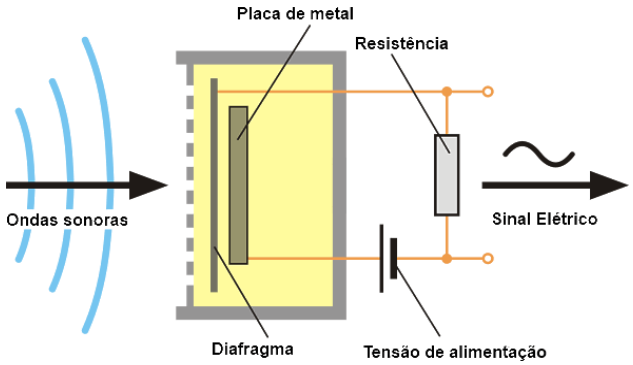

Fig. 15 - Esquema ilustrativo do microfone Electret (fonte http://www.holmco.de/mik.html, modificada)

O seu princípio de funcionamento está relacionado com a pressão sonora que incide sobre a membrana do diafragma, fazendo-a vibrar e alterar o espaço existente em relação à placa de metal. Isto modifica a forma do condensador que por consequência faz variar a sua capacitância, causando variações de tensão proporcionais à oscilação da onda de som. De uma forma resumida, um microfone capacitivo converte a distância entre as placas do condensador numa tensão elétrica. Este fenómeno que se pode compreender através da equação (9):

$$V = q\frac{d}{A \cdot \varepsilon_0} \tag{9}$$

Onde:

V = Tensão (V)
q = Carga do condensador (C)
d = Distância entre o diafragma e a placa de metal (m)
A = Área da membrana do diafragma (m^2)
ε_0 = Constante dielétrica (8,854 × 10^{-12} F/m)

Normalmente um microfone produz sinais elétricos com uma amplitude bastante reduzida, por isso não pode ser ligado diretamente a um microcontrolador, necessitando o sinal de ser amplificado. Esta placa desenvolvida pela Sparkfun, já tem integrado, no circuito, um amplificador operacional (OPA344) que amplifica o sinal de modo a possibilitar a sua leitura por parte do Arduino. Para além disso o circuito integrado tem um divisor de tensão, que faz um *offset* no sinal de saída no valor de 1,65 volts, para que toda a onda tenha valores positivos. Na Fig. 16 pode observar-se uma imagem do sensor de ruído utilizado neste trabalho.

Fig. 16 - Sensor de ruído (Placa de circuito impressa para o Electret microfone) (fonte: http://www.spikenzielabs.com)

Principais características (Knowles Acoustics, 2004):

- Alimentação: 3,3 V
- Ligações: Vcc, GND, A0
- Sensibilidade: -44 dB re 1V/Pa
- Direccionalidade: OMNI-DIRECIONAL
- Temperatura de operação: -25 a +55 ºC
- Dimensões: 17 × 10 mm

4.2.8 Relógio de tempo real (RTC)

Para o projeto foi escolhido como RTC o modelo DS1307 que é produzido pela Dallas Semicondutor. O relógio de tempo real é um dispositivo que tem a capacidade de contar o tempo de forma precisa, fazendo, para isso, uso da oscilação de um cristal incorporado, neste caso de 32,768 kHz (Semicondutor, 2009). Este mantem-se ativo mesmo que a alimentação esteja desligada, pois possui uma bateria de apoio, que lhe permite uma contagem contínua do tempo, por um período de pelo menos 9 anos, sem alimentação externa.

O RTC tem uma interface I2C para possibilitar a transferência de dados através deste protocolo e fornece relógio e calendário completo. Ou seja, disponibiliza contagem de segundos, minutos, horas, dias, dias da semana, mês e ano. Tem inclusivamente correções para os meses com menos de 31 dias e anos bissextos. A precisão do dispositivo depende da precisão do cristal e podem ocorrer erros adicionais devido a desvios na frequência do cristal, provocados por alterações de temperatura ou influência de um campo magnético. A sua flutuação de tempo pode ir até 1 minuto por mês. O relógio de tempo real tem integrado um circuito de controlo do fornecimento de energia que quando deteta

falha na alimentação principal, passa automaticamente para a bateria de apoio. Na Fig. 17, pode ver-se uma imagem do RTC.

Fig. 17 – Relógio de tempo real frente e verso

Principais características:

- Alimentação: 5 V
- Ligações: Vcc, GND, pino A4 (SDA) e A5 (SCL)
- Interface de hardware: I2C
- Calendário preciso até 2100
- Memória NV SRAM: 56 Bytes
- Relógio/calendário de código binário decimal (BCD)
- Consumo: <500 nA em modo de bateria
- Dimensões: 20 × 20 mm

Este dispositivo traz bastantes vantagens ao sistema pois elimina inconvenientes, como o facto de a contagem do tempo ser interrompida quando o sistema é desligado, não sendo, assim, necessário reiniciá-la quando se volta a alimentar o mesmo. Isto acontece pois o dispositivo continua a funcionar mesmo sem alimentação externa, permitindo uma contagem contínua do tempo. Este componente permite acionar a recolha de dados segundo os intervalos de tempo pretendidos e, para além disso, registar a hora e data em que foi adquirida a informação dos sensores, tudo isto sincronizado com o tempo real.

4.3 Software

4.3.1 Processing

O Processing é um software *open source*, desenvolvido pelo Massachusetts Institute of Technology (MIT), que pode ser utilizado com a finalidade de criar interações gráficas de visualização no computador. O seu ambiente de desenvolvimento (PDE – Processing Development Environment) foi construído com base na linguagem de programação Java e os programas criados denominam-se *sketches*. Disponibiliza várias bibliotecas, já integradas, que aumentam as suas capaciadades e possiblilita aos utilizadores, também, a criação de novas bibliotecas (Karvinen, et al., 2011).

Com esta plataforma de software, os dados provenientes da placa Arduino podem ser processados e tratados de forma mais complexa, ou seja, é possível criar uma interface gráfica para visualização dos mesmos, guardá-los, e até reportá-los para a nuvem de computação/internet (quando já estabelecida a ligação).

Para interagir com a placa Arduino, existem bibliotecas criadas com esse fim, e que permitem receber informação ou mesmo enviar comandos. Normalmente isto é feito por via de comunicação serie mas também se pode efetuar usando comunicação sem fios.

Neste trabalho, o software em questão é bastante útil, pois permite o desenvolvimento da interface gráfica do sistema de desenvolvido e possibilita a criação de um servidor que monitoriza a porta local para onde a Wifly envia os dados, recolhendo-os e fazendo o seu tratamento para amostragem gráfica. É ainda responsável por enviar os dados recolhidos para a internet.

4.3.2 Arduino IDE

O ambiente de desenvolvimento Arduino IDE (Arduino Integrated Development Environment) é uma aplicação que comporta todo o software necessário para fazer a interação da placa eletrónica Arduino com o computador, de modo a poder programar o microcontrolador. Foi desenvolvido com base no Processing PDE, sendo por isso bastante similar. Estas duas aplicações de software são passíveis de utilizar em conjunto, ao mesmo tempo que se trabalha com a placa Arduino, podendo as três ferramentas interagir facilmente entre si. Por ser inspirado no projeto Processing, o Arduino IDE também é escrito em Java e utiliza o paradigma de programação estruturada. Este software inclui uma biblioteca Wiring, que permite utilizar uma linguagem de código baseada em C/C++ e através dele é possível criar, editar, compilar e guardar programas de código, denominados *sketches*, que são carregados para a placa dando-lhe instruções. O Arduino IDE tem, ainda, integrada uma interface gráfica, chamada Serial Monitor, para monitorizar as comunicações USB do Arduino (em formato de texto), e permite também o envio de informação no sentido contrário. Este software dispõe de várias bibliotecas que facilitam o trabalho do utilizador, pois abstraem a complexidade inerente aos componentes de hardware, nomeadamente ao nível da comunicação (Doukas, 2012).

4.4 Protocolos de Comunicação

4.4.1 USB

A comunicação USB (Universal Serial Bus) permite fazer a transferência de informação entre a placa Arduino e outros dispositivos, como o computador, e, além disso pode fornecer alimentação. Este tipo de comunicação também pode ser usado para estabelecer uma interação entre o computador e sensores (como no caso deste projeto), ou outros dispositivos que estejam conectados ao Arduino.

O Arduino Uno, como referido anteriormente, está munido de uma conexão com tecnologia USB, para poder comunicar com outros dispositivos de hardware.

O Arduino IDE disponibiliza uma biblioteca, já integrada, que abstrai a complexidade do hardware, facilitando assim a utilização deste método de comunicação. Por seu turno, o Serial Monitor permite visualizar os dados a serem enviados ou recebidos na porta USB. Também o software Processing tem bibliotecas (*Arduino* e *Serial*) que possibilitam receber e enviar dados por USB de modo a interagir com a placa Arduino.

Grande parte do trabalho desenvolvido com o Arduino neste projeto foi realizado através da utilização da porta serie para descarregar os *sketches*, de forma a poder testar ou executar o código desenvolvido.

4.4.2 I2C

O protocolo I2C (Inter-Integrated Circuit) é uma tecnologia de comunicação que foi concebida para facilitar e padronizar a troca de informação entre sensores e microcontroladores, como é o exemplo do Atmega328. É especialmente utilizada para sensores que não necessitam de transferir uma grande quantidade de dados, pois estes não podem circular nos dois sentidos em simultâneo, o que torna a ligação mais lenta. O seu princípio de funcionamento baseia-se num barramento com duas ligações: uma para transferir dados, o fio SDA (Serial Data Line), e outra que funciona como relógio, o fio SCL

(Serial Clock Line), fazendo o compasso de espera entre cada transferência de informação, para assegurar que o barramento só processa dados de um sensor de cada vez. A este barramento podem ser conectados múltiplos sensores (até 127 unidades), os denominados *slaves*, que partilham os fios SCL,SDA, massa e alimentação (no caso de esta ser a mesma), e existe sempre um dispositivo que coordena a comunicação entre estes, o *master* (Margolis, 2011).

O Arduino Uno utiliza o pino analógico 5 para o SCL, que fornece um sinal de relógio, e o pino analógico 4 para o SDA. Na Fig. 18 está apresentada, através de um esquema, a exemplificação de uma ligação de barramento I2C com um master (ex. Arduino Uno) e vários *slaves* (ex. sensores).

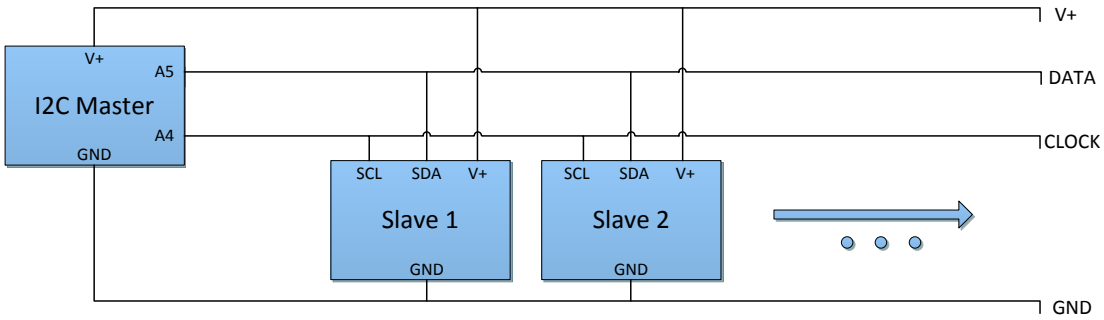

Fig. 18 - Esquema de barramento de ligação I2C entre o master e múltiplos slaves

Cada sensor deverá possuir um endereço diferente, que o identifica durante a comunicação. Dependendo do dispositivo, em relação ao endereço estes podem apresentar as seguintes características:

- ter um endereço fixo;
- permitir que se configure o endereço através da ligação, ou não, de um fio específico;
- necessitar de acionamento de comandos de iniciação;

Para poder fazer a comunicação I2C através de código é necessário analisar bem a informação proveniente na ficha do produto, onde consta, normalmente, o endereço I2C ou opções de escolha do mesmo, a tensão de operação, a explicação de ligação do circuito e também indica os passos e comandos para ativação do produto e leitura de dados.

No presente trabalho, a ligação dos sensores TSL2561, K30 e do RTC é feita através da utilização deste protocolo, sendo que o Arduino Uno desempenha a função de master. O circuito utilizado para ligar o sensor SHT15 assemelha-se bastante a um barramento I2C, no entanto tem algumas diferenças.

4.4.1 SPI

O SPI (Serial Peripheral Interface) é outro modelo de barramento do género I2C mas tem uma taxa de processamento de dados mais elevada (Margolis, 2011). Possibilita o envio e receção de dados em simultâneo, pois as suas ligações de entrada e saída estão separadas. Este tipo de comunicação faz uso de uma conexão adicional por cada dispositivo ligado, o que acarreta um maior número de conexões, no caso de se pretender ligar um maior número de dispositivos. Neste protocolo a ligação de entrada (MOSI), de saída (MISO) e de sinal de relógio (SCLK) estão separadas. Os *slaves* são identificados pela ligação Slave Select (SS), pino 10 do Arduino Uno.

A comunicação SPI é usada neste trabalho para fazer a conexão com o cartão micro-SD, onde as ligações MOSI, MISO e de SCLK são feitas pelos pinos digitais 11, 12 e 13, respetivamente. O pino SS, neste caso denominado CS é o pino digital 4 da Wireless SD Shield, como referido anteriormente.

4.4.2 Wifi

Existem várias formas de comunicação sem fios, no entanto a usada para o desenvolvimento deste sistema foi o Wifi. Esta permite a transferência de dados entre equipamentos remotamente, em curtas distâncias, através da utilização de um sinal rádio. O Wifi é um protocolo de comunicação internacional desenvolvido e certificado pela Wi-Fi Aliance para definir as tecnologias de redes locais sem fios (WLAN – Wireless Local Area Networks) implementadas pela norma 802.11, que é definida pela IEEE (Institute of Electrical and Electronics Engineers). Este protocolo funciona na banda de frequência 2,4 GHz e estabelece normas para a criação e uso de redes sem fios que assim podem funcionar como uma ligação Ethernet, mas sem fios. É normalmente utilizado para aplicações onde é necessário estabelecer ligação com a internet. Wifi foi o nome estabelecido para a certificação dada pela Wi-Fi Alliance à compatibilidade entre dispositivos que comunicam utilizando o padrão 802.11.

São descritos vários padrões da família IEEE 820.11, sendo que os mais utilizados e mais conhecidos atualmente são os 802.11b, 802.11g e 802.11n (Telecom Regulatory Authority, 2003).

A tipologia de uma rede que respeite a norma IEEE 802.11, consiste numa arquitetura celular, constituída por células ou estações (STA - Station) - dispositivos que acedem à rede ou clientes - e pontos de acesso (AP – Access Point) - dispositivos que operam como uma estação base para fornecer o acesso à rede. Uma rede é estabelecida quando, uma ou mais STAs se ligam a um ponto de acesso (ver Fig. 19). Cada rede sem fios deve ser distinguida com uma identificação própria, a SSID (Service Set Identifier).

Fig. 19 – Arquitetura de rede Wifi (estações ligadas a um ponto de acesso)

No que diz respeito à segurança da rede é necessário haver precauções. A sua identificação é um primeiro passo mas não é suficiente. Assim, a segurança de uma rede sem fios é feita através de dois processos, chamados Encriptação e Autenticação. A Autenticação é o processo através do qual se certifica que cada estação tem autorização para comunicar com outra estação na mesma rede ou com um ponto de acesso. A Encriptação aplica algoritmos para encriptar os dados enviados entre estações, prevenindo que intrusos intercetem a comunicação. Existem três diferentes mecanismos de segurança que têm sido desenvolvidos ao longo dos anos, o WEP, o WPA e o WPA2, por ordem crescente de nível de segurança. Estes mecanismos funcionam através de chaves de segurança que são solicitadas aos clientes/estações para que estes se possam conectar à rede.

O Wifi, tem diversas vantagens, como o facto de facilitar o acesso a uma rede e a inserção de outros dispositivos, sem a necessidade de usar cabos, e permitir o acesso em qualquer local dentro dos limites de alcance da transmissão (cerca de 100m), aumentando a mobilidade. O alcance pode variar com o espaço onde se encontram os APs e os obstáculos existentes entre estes e os STAs.

A comunicação Wifi apesar de ser mais robusta em relação a outros protocolos de comunicação, como o *Bluetooth* e o *Zigbee*, apresenta, também, um consumo de energia superior.

Uma vez que o módulo Wifly não suporta o modelo de segurança utilizado pela *eduroam* não foi possível estabelecer a ligação sem fios à rede sem fios da Universidade de Lisboa. Assim, para estabelecer a comunicação remota do sistema desenvolvido (ALVI), foi necessário criar uma rede Wifi (denominada Sensor) através de um router, que funciona como ponto de acesso. Assim, é possível a transferência de dados do sistema de monitorização para um computador desde que o módulo Wifly e o computador (estações) estejam ligados à mesma rede, neste caso a rede Sensor.

A transferência de informação é feita com base no uso de um conjunto de protocolos para troca de dados sequenciais entre computadores em ambientes de redes locais ou remotas, chamado TCP/IP.

4.5 Internet of things (IoT)

A *Internet of Things* ou "Internet das Coisas" é um novo conceito que visa juntar várias tecnologias para interligar sensores e atuadores com a internet. Apesar de ter uma definição mais complexa pode ser descrita como uma junção entre a computação física e a computação em nuvem. Ou seja, permite criar sistemas capazes de recolher dados, monitorizar o ambiente físico e, além disso, comunicar com a internet, armazenando e partilhando a informação online, em diferentes interfaces (ver Fig. 20).

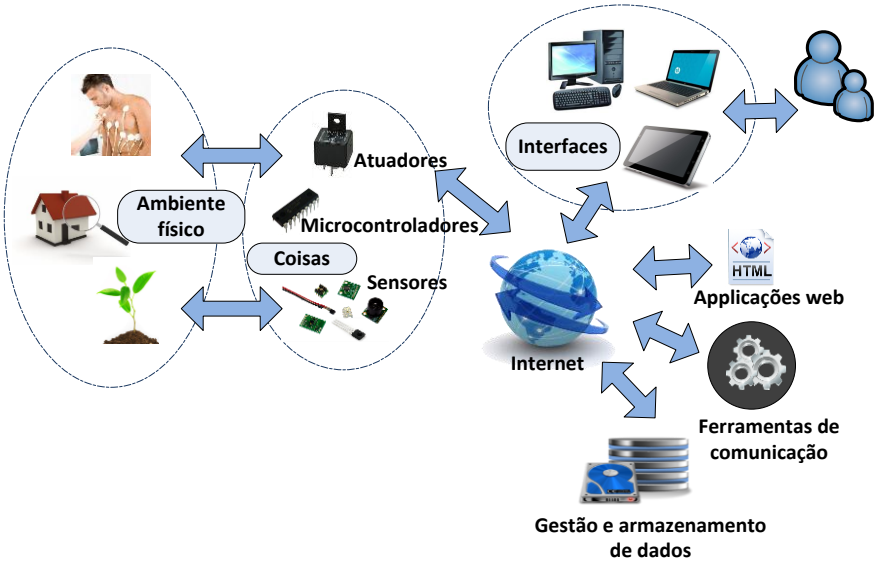

Fig. 20 – Esquema representativo da *Internet of Things* (baseado em imagem de Doukas, 2012)

Neste contexto, podem encontrar-se algumas plataformas de suporte para gestão e visualização dos dados online e em tempo real. Estas são disponibilizadas para uso gratuito e foram criadas com este propósito, no conceito da *Internet of Things*. Dentro das opções existentes, a mais conhecida é o Cosm, que se apresenta como uma boa solução para o presente projeto e é já utilizada em projetos semelhantes. A sua interface mostra a informação enviada através de gráficos para uma melhor interpretação dos dados.

4.6 Montagem do protótipo

De seguida será abordada toda a montagem do sistema, incluindo a parte do hardware e do software. Em relação ao hardware serão mostrados esquemas para explicar as ligações dos componentes e no que se refere ao software será feita a apresentação dos algoritmos, sob a forma de fluxogramas de código, de modo a elucidar sobre o código produzido para o funcionamento do sistema.

4.6.1 Hardware

Na montagem do protótipo foram conectados todos os componentes, de acordo com a sua interface de hardware e as suas ligações de alimentação. No esquema eletrónico apresentado na Fig. 21 pode observar-se, de uma forma global, como foram feitas estas ligações entre os sensores escolhidos e o Arduino Uno, através da Wireless SD Shield a ele acoplada. De seguida são explicadas em maior detalhe essas ligações.

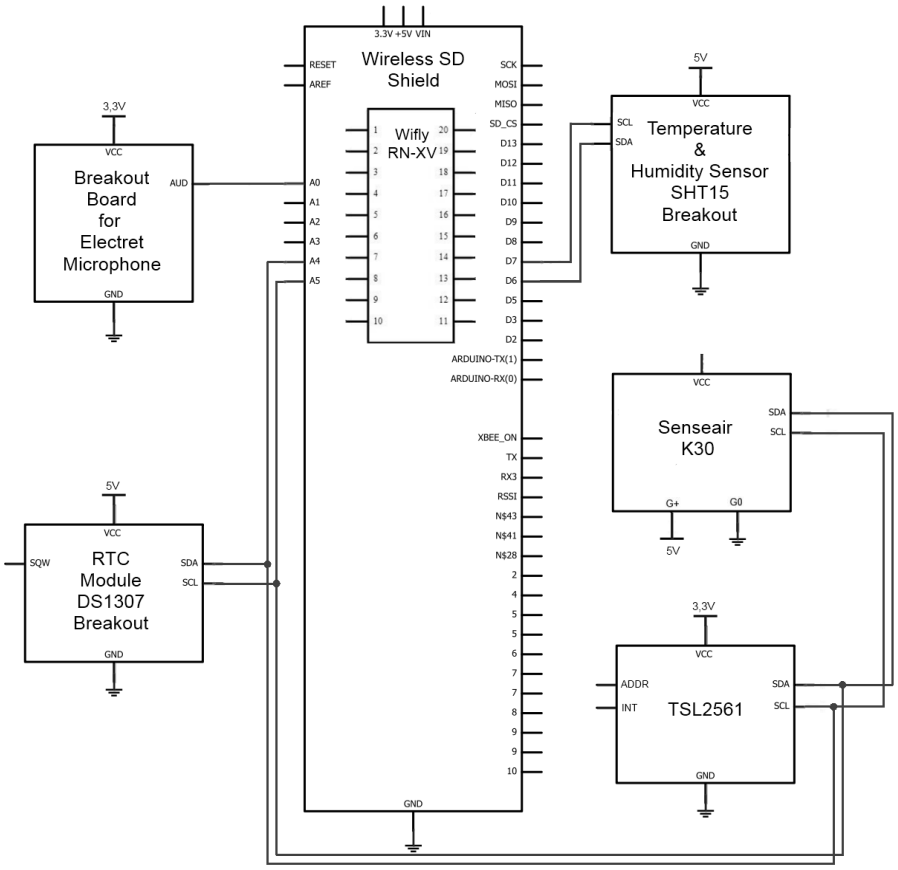

Fig. 21 - Esquema eletrónico do sistema de monitorização desenvolvido (Fritzing)

Sensor SHT15

Na ligação deste sensor com o Arduino Uno, usam-se os pinos digitais 6 (SCK) e 7 (DATA), 5V para a alimentação e por último a massa (GND). Como referido na descrição deste componente, a sua ligação é feita com um protocolo semelhante ao I2C. Na, Fig. 21 pode ver-se o esquema de ligação do SHT15 com a placa Arduino.

Sensores TSL2561, K30 e RTC

Estes três dispositivos, como já foi referido, têm uma interface de hardware que pode ser feita pelo protocolo I2C, por isso a sua ligação foi estabelecida através de um barramento I2C onde compartilham o fio da massa (GND) e os 2 fios de SCL e SDA. No que diz respeito à alimentação, esta não pode ser partilhada, pois o TSL2561 deve ser alimentado a 3,3V, enquanto que, o K30 e o RTC são alimentados por uma tensão de 5V, devido aos requisitos de cada circuito.

Ao nível da comunicação I2C surgiu uma barreira relacionada com a incompatibilidade dos endereços do Senseair K30 e do RTC, pois têm endereços de fábrica iguais e isso não permite o funcionamento correto deste protocolo de comunicação. Neste caso, foi necessário analisar a comunicação de ambos os dispositivos e verificou-se que o RTC tem endereço fixo, mas o Senseair K30 é suscetível de configurar. Assim, após uma pesquisa, procedeu-se à reconfiguração do sensor para um novo endereço, recomendado pelo fabricante.

Microfone Electret

O sensor de ruído, microfone Electret, foi ligado ao Arduino Uno utilizando o V+, massa e o pino analógico A0, sendo que o último recebe os valores de tensão provenientes do canal AUD do sensor, correspondentes às contagens ADC da pressão sonora recebida. Estes valores analógicos de 0 a 5V são digitalizados pelo Arduino Uno para valores entre 0 a 1023 (10 bits).

Wireless SD Shield e módulo Wifly

A Wireless SD Shield é, como referido na sua descrição, uma placa adicional desenvolvida propositadamente para ligar ao Arduino Uno, sendo que os pinos de ligação de ambos encaixam perfeitamente. O mesmo acontece com o módulo Wifly na sua conexão à Wireless SD Shield. Estes dois componentes comunicam com o Arduino Uno através dos pinos RX e TX.

Na Fig. 22 pode observar-se o aspeto final da estrutura e design do ALVI, o sistema de monitorização remota de condições de conforto interior desenvolvido neste trabalho.

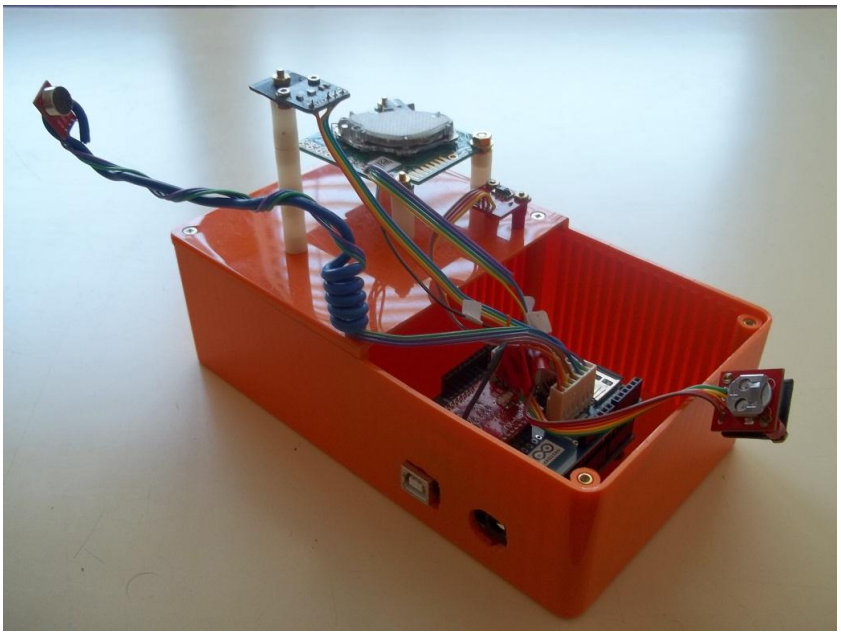

Fig. 22 – Aspeto final do ALVI

4.6.2 Código Arduino

Para que toda a eletrónica do sistema desenvolvido possa funcionar, além de ser necessário que todos os constituintes estejam corretamente ligados e alimentados, é preciso interagir com o hardware através de software. É neste aspeto que o Arduino Uno se apresenta como o componente mais importante do sistema, pois é este que coordena todas as comunicações com os restantes dispositivos. Assim, foi desenvolvido no software Arduino IDE um programa que define todas as instruções para o Arduino Uno, para que este possa enviar os comandos necessários a cada componente, sejam eles de ativação ou de solicitação de serviços (ex. leitura dos dados recolhidos).

O programa elaborado (ver Anexo C), cujo fluxograma de código se apresenta Fig. 23, é constituído, essencialmente, por duas rotinas principais o *setup* e o *loop*, que caracterizam um *sketch* do Arduino IDE.

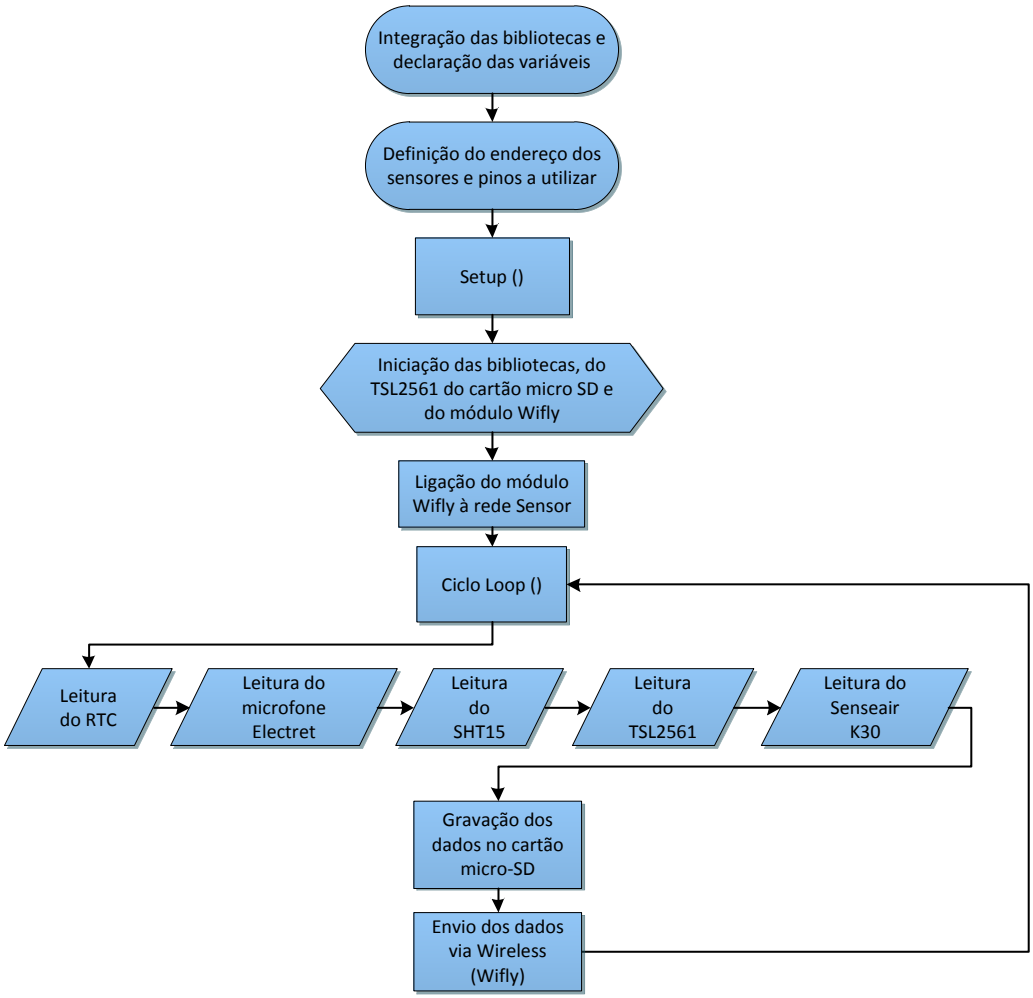

Fig. 23 - Fluxograma de código do *sketch* do Arduino IDE desenvolvido

A parte inicial do *sketch* consiste, essencialmente, na integração das bibliotecas a ser utilizadas, n declaração de variáveis necessárias (credenciais da rede Sensor, variáveis para guardar a leitura dos sensores e efetuar cálculos) e na definição do endereço dos sensores que partilham o barramento I2C e dos pinos do Arduino a ser utilizados. Neste caso incluíram-se as bibliotecas "Wire" (para a comunicação I2C), "SD" (para comunicação com o cartão micro-SD), "TSL2561" (biblioteca do sensor de intensidade luminosa) e "WiflyHQ" (biblioteca do módulo Wifly).

Na função *setup*, rotina que corre apenas uma vez, são iniciadas as bibliotecas referidas, estabelece-se a comunicação com o cartão micro-SD, configura-se o TSL2561 e enviam-se os comandos necessários para iniciar o módulo Wifly, dando-lhe instruções para se ligar à rede Sensor.

Na função *loop*, ciclo que se repete continuamente, é efetuada a leitura do relógio de tempo real e a do sensor microfone Electret, a cada minuto faz-se a leitura dos sensores SHT15, TSL2561 e Senseair K30, gravam-se os dados no cartão micro-SD (ficheiro CSV) e são transmitidos remotamente pelo módulo Wifly para a porta local 1234 (através da rede Sensor). Devido a um erro que não foi possível ultrapassar, o ficheiro CSV não é gravado com um cabeçalho a descrever as variáveis guardadas automaticamente, sendo necessário acrescentar manualmente. É portanto importante referir que da esquerda para a direita os dados são gravados no ficheiro pela seguinte ordem: hora e data da medição, temperatura (ºC), humidade relativa (ºC%), concentração de CO_2 (ppm), iluminância (lux) e ruído (médias a cada minuto dos valores RMS das contagens ADC).

4.6.3 Código Processing

A aplicação para a interface do sistema foi desenvolvida na sua totalidade com a utilização do software Processing (ver Anexo D). Esta faz um tratamento da informação e disponibiliza ao utilizador uma visualização e avaliação em tempo real das condições de ambiente interior existentes. Para a sua construção desenvolveu-se um programa no Processing PDE, que pode ser ilustrado pelo fluxograma da Fig. 24.

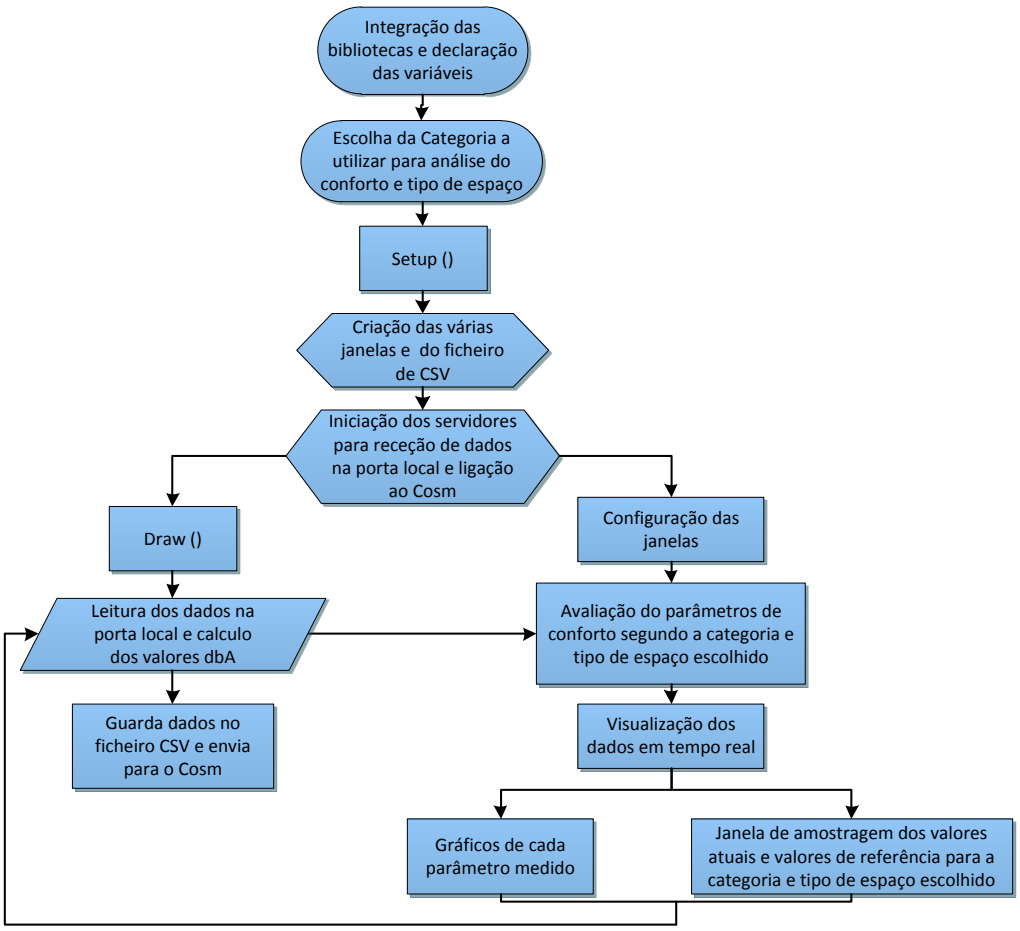

Fig. 24 - Fluxograma do programa desenvolvido para construção da interface

Interface

O interface tem por objetivo proporcionar uma simples, fácil e rápida visualização do estado atual da qualidade do ambiente interior. A informação apresentada deve ser, por isso, restringida apenas aos aspetos mais relevantes. A parte estética foi também tida em conta, tentando criar um aspeto visual apelativo para que seja de alguma forma atrativo para o utilizador. Para isto elaborou-se uma janela onde são apresentados os gráficos dos vários parâmetros em tempo real, assim como os valores correspondentes, acompanhados dos valores de referência para cada categoria. No que diz respeito aos gráficos, são apresentados dois em simultâneo, existindo a possibilidade de alternar entre os cinco parâmetros de qualidade ambiental tanto para o gráfico superior como para o inferior. Isto torna possível comparar a evolução de cada variável no tempo e observar possíveis influências existentes entre elas.

A transferência de informação entre o Arduino e o software da interface criada é feita através da rede Sensor. Para isso é necessário a criação de um servidor no Processing que fica à espera da chegada de informação na porta 1234, informação esta que será enviada pela Wifly a partir do momento em que este dispositivo estabelece a ligação com o servidor. Para isto é também necessário um router que faz a ponte entre a Wifly e o computador, estabelecendo a rede Wifi local (Sensor). Assim, através do servidor são importados no Processing os dados da porta local 1234.

No código criado, cujo diagrama de fluxo se apresenta na Fig. 24, após a aquisição dos dados, é implementado um modelo de decisão em tempo real. Neste modelo os dados são tratados, analisados e definem-se os valores de referência correspondentes, que são recomendados pelas normas existentes para cada categoria de QCI e tipo de espaço. De seguida é efetuada uma avaliação da qualidade do ambiente interior através da comparação do valor atual com os valores limite referência (VRmax e VRmin). Os intervalos utilizados para classificar a QCI foram definidos com critérios que se consideraram plausíveis, tendo em conta a ordem de grandeza dos valores de cada parâmetro (ver Tabela 9).

Tabela 9 - Critérios de avaliação em tempo real da QCI

Parâmetro	Critério	Resultado
Temperatura	(VRmin + 1) < Temperatura < (VRmax - 1)	QCI adequada
	(VRmin - 1) ≤ Temperatura ≤ (VRmax +1)	QCI aceitável
	(VRmax + 1) < Temperatura < (VRmin - 1)	QCI inadequada
Humidade Relativa	(VRmin + 2) < HR < (VRmax - 2)	QCI adequada
	(VRmin -2) ≤ HR ≤ (VRmax + 2)	QCI aceitável
	(VRmax + 2) < HR < (VRmin - 2)	QCI inadequada
Concentração de CO_2	0 ≤ Concentração de CO_2 < (VRmax - 100)	QCI adequada
	(VRmin - 100) ≤ Concentração de CO_2 ≤ (VRmax +100)	QCI aceitável
	Concentração de CO_2 ≥ (VRmax + 100)	QCI inadequada
Iluminância	Iluminância ≥ VRmin	QCI adequada
	VRmin*0,67 ≤ Iluminância ≤ VRmin	QCI aceitável
	Iluminância ≤ (VRmin*0,67)	QCI inadequada
Nível de ruído	Nível de ruído < (VRmáx – 2)	QCI adequada
	(VRmax – 2) ≤ Nível de ruído ≤ (VRmin + 2)	QCI aceitável
	Nível de ruído > (VRmáx + 2)	QCI inadequada

De modo a tornar a compreensão e observação do modelo mais intuitiva e evidente foi introduzido um esquema tipo semáforo, que estabelece o seguinte padrão:

- Verde – Qualidade do ambiente interior adequada
- Amarelo – Qualidade do ambiente interior aceitável
- Vermelho – Qualidade do ambiente interior inadequada

Esta apreciação é feita de forma isolada para cada um dos critérios de qualidade definidos, não havendo qualquer análise global. O que se deve ao facto de ser bastante difícil ajuizar qual o peso de cada parâmetro na qualidade do ambiente interior como um todo. Na Fig. 25 apresenta-se o aspeto da janela criada para interface do sistema de monitorização.

O software de interface permite ainda a gravação dos dados provenientes dos sensores num ficheiro do tipo CSV, assegurando assim mais uma forma de gravação local dos dados que reduz a volatilidade da simples visualização. Esta forma de armazenamento está limitada pela fidelidade da rede Wifi, o que não acontece no caso da gravação de dados diretamente no cartão micro-SD.

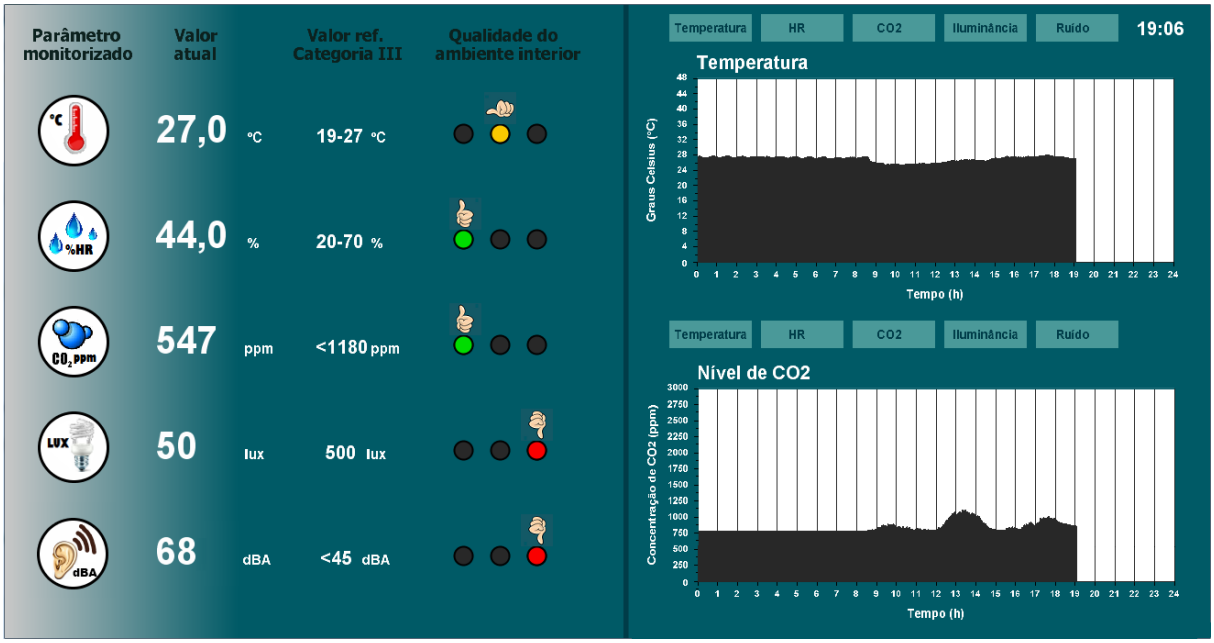

Fig. 25 - Interface do sistema construído para visualização no computador

4.6.4 Suporte de informação online (Cosm)

O suporte de informação online no site do Cosm foi conseguido através da utilização de uma biblioteca do Processing e da integração, no *sketch*, do código necessário para exportar os dados. Esta mesma biblioteca denomina-se "Cosm" e foi criada com esse fim.

No site do Cosm é necessário fornecer o IP do computador de onde se pretende enviar os dados para que seja possível a conexão. É necessário referir que não é possível fazer esta ligação utilizando o IP de um computador que esteja ligado à rede eduroam, pois os IP´s desta rede são privados, não possibilitando dessa forma este tipo de acesso externo.

A rede sem fios Sensor, utilizada para a comunicação remota é uma rede local, através da qual não é possível aceder à internet. Por essa razão é necessário que o computador onde se visualiza a interface esteja ligado a cabo Ethernet com acesso à internet, de modo a que seja possível transmitir a informação para o Cosm.

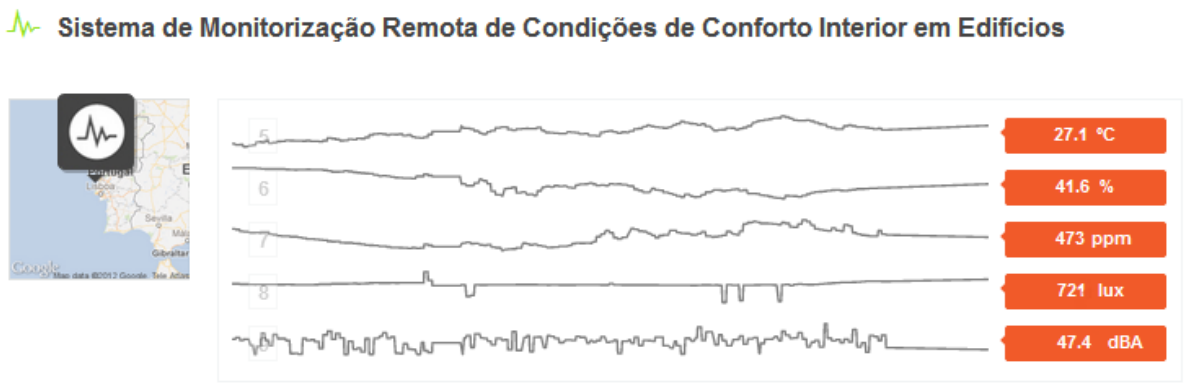

Fig. 26 – Interface do suporte de dados online (Cosm)

4.7 Calibração do sistema

Com o intuito de fazer uma validação dos dados produzidos pelo sistema, realizou-se uma medição de calibração para comparar os dados obtidos pelo protótipo desenvolvido com as medições de sensores calibrados. Deste modo colocaram-se todos os sensores no mesmo local, o mais próximos possível para que estivessem a medir nas mesmas condições e efetuou-se uma medição durante duas horas com uma taxa de amostragem de 1 minuto. Este teste foi feito no bar do C1 durante a hora de almoço (das 12h às 14h), pois neste local e a esta hora é expectável a existência de variações dos parâmetros medidos o que possibilita uma melhor verificação da qualidade dos dados.

No caso do sensor de ruído, este teve de ser caracterizado por falta de informação disponível, por isso foram realizados mais testes para além deste, igualmente descritos adiante.

Apesar de a medição ter sido realizada com todos os sensores em simultâneo, a análise dos resultados é feita para cada sensor individualmente.

4.7.1 Sensor de temperatura e humidade

Na calibração do sensor SHT15 foi utilizado o dispositivo IAQ-Calc 754 (ver Fig. 27).

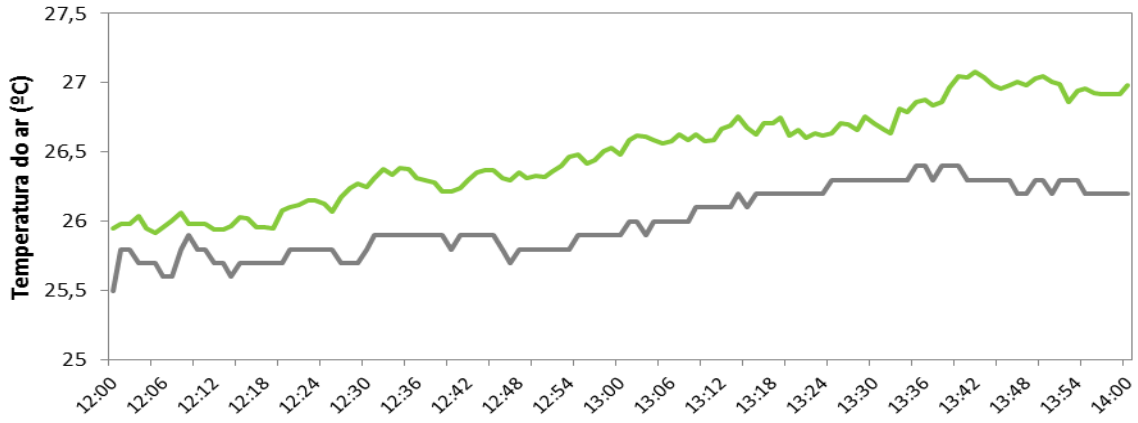

Fig. 27 – Gráficos dos valores obtidos para a temperatura do ar pelos dois sensores SHT15 (linha a verde) e IAQ-Calc 7545 (linha a cinzento)

Para a medição da temperatura do ar, como pode verificar-se pelo gráfico da figura acima, a resposta de ambos os sensores é semelhante, pois as curvas apresentam variações nos dados recolhidos bastante parecidas e os valores produzidos estão bastante próximos com um desvio médio de apenas

2%. Sabendo que o erro do sensor IAQ-Calc para a temperatura é de ±0,6 ºC e que o erro do SHT15 é de ±0,3 ºC, pode considerar-se que para este parâmetro as medições do SHT15 são bastante satisfatórias.

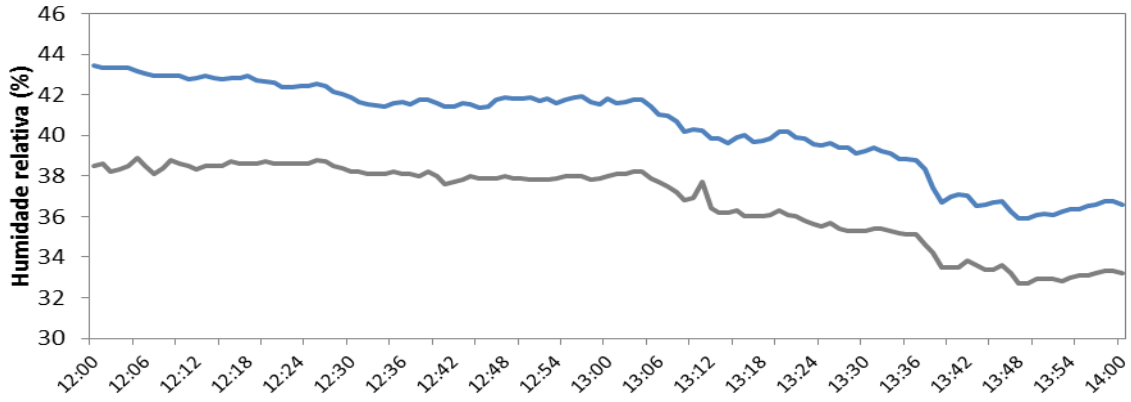

Fig. 28 - Gráfico dos valores medidos pelo sensor SHT15 (linha a azul) e pelo sensor de humidade relativa do IAQ-Calc 7545 (linha a cinzento)

Na medição da humidade relativa, como é possível observar na Fig. 28, o sensor SHT15 efetua medições de valores bastante aproximados aos do sensor existente no IAQ-Calc 7545, apresentando estes um desvio médio de 10%. A evolução do gráfico dos dois sensores para este parâmetro tem um comportamento idêntico, apresentando um aumento e redução dos valores nos mesmos períodos. Tendo em conta que o erro do SHT15 é ±2% e o erro do sensor de humidade do IAQ-Calc é de ±3%, pode dizer-se que os sensores produzem medições com um desvio maior do que o esperado, no entanto os valores estão na mesma ordem de grandeza.

Dados estes resultados para ambos os parâmetros medidos pelo sensor pode, assim aferir-se que qualidade dos dados medidos é aceitável pois, faz medições na mesma ordem de grandeza do sensor referência e com um erro reduzido. No entanto, este resultado é menos satisfatório que o esperado pois o SHT15 é um sensor já calibrado e com boa precisão, de acordo com o fabricante.

4.7.2 Sensor de CO_2

A calibração do Senseair K30 foi efetuada também com o auxílio do IAQ-Calc 7545, por comparação com dos valores medidos pelos dois sensores (ver Fig. 29).

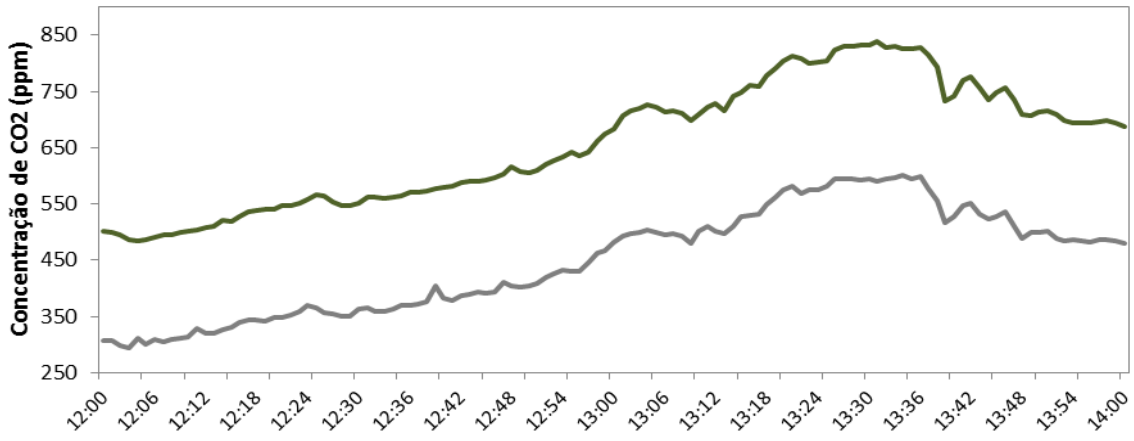

Fig. 29 - Valores obtidos pelo Senseair K30 (linha a verde) e o sensor de CO_2 do aparelho IAQ-Calc (linha a cinzento) no teste de calibração

O gráfico da figura acima mostra que acurva dos dois sensores é muito similar, sendo que os picos de ambos coincidem e estão sincronizados, portanto a resposta dos sensores é idêntica. No entanto, os valores dos resultados obtidos estão um pouco afastados daquilo que se esperava, ou seja, o desvio médio entre os valores dos sensores é de 45%, um valor relativamente elevado. É um resultado que não era expectável pois o Senseair K30 trata-se de um dispositivo que está devidamente calibrado, de acordo com o fabricante. Este resultado também não pode ser explicado pelo erro dos sensores que é de ±30 ppm, acrescido de 5% do valor medido para o Senseair K30 e de ±50 ppm para o sensor de referência.

No entanto os valores obtidos pelo IAQ-Calc também não estão dentro dos valores normais para um espaço interior, já que descem abaixo dos valores médios da concentração de CO_2 na atmosfera exterior, ou seja, mediu valores abaixo dos 300ppm.

Após verificação do certificado de calibração do dispositivo IAQ-Calc 7545 utilizado como sensor referência, chegou-se à conclusão que este aparelho deve fazer calibrações periódicas e já ultrapassou a data de revisão há bastante tempo (em falta desde 2009), portanto necessita de ser calibrado. Este facto não valida a qualidade dos dados obtidos pelo Senseair K30, no entanto, estes são plausíveis pois encontram-se dentro dos valores normais e além disso este é um sensor calibrado de forma exigente, segundo o fabricante. Posto isto, considera-se que o sensor produz dados de qualidade e que estão de acordo com a realidade.

4.7.3 Sensor de intensidade luminosa

No que concerne ao sensor TSL2561, a sua calibração foi feita com o Luxímetro Extech 401036 (ver Fig. 30).

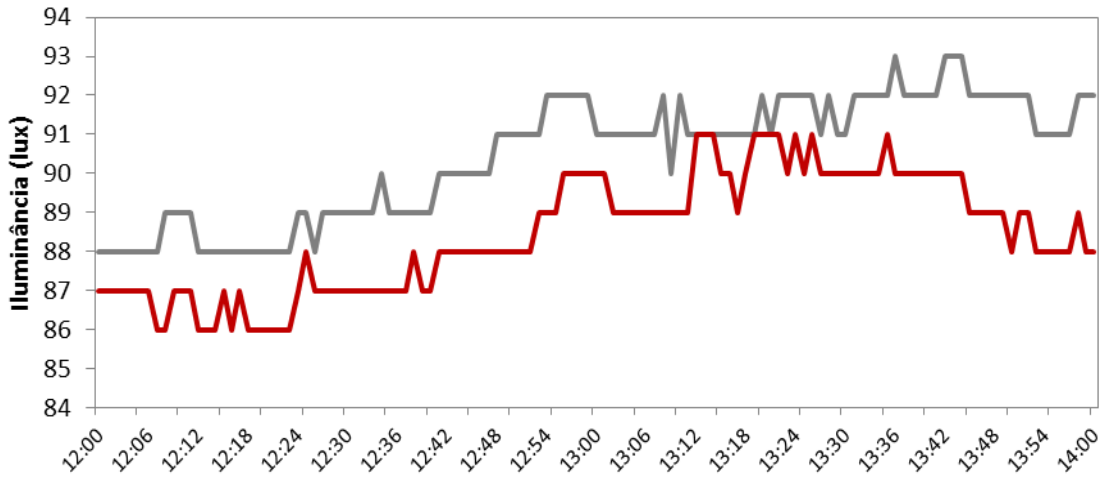

Fig. 30 – Gráfico dos valores obtidos durante o teste de calibração para os sensores TSL2561 (linha a vermelho) e Extech 40136 (linha a cinzento)

Como pode ser observado na figura acima, os valores obtidos pelos dois sensores estão bastante próximos, com um desvio médio entre eles na ordem dos 2%, o que se considera satisfatório. A resposta dos sensores também é semelhante, pois excetuando alguns pontos específicos, as curvas têm comportamentos muito parecidos e sincronizados. Em relação aos erros dos sensores, o fabricante do TSL2561 não disponibilizou informação sobre este aspeto e o Exctech 401036 apresenta um erro de ±3% na leitura. Pode assim dizer-se que o TSL2561 recolhe dados com boa qualidade, pois se for considerado o erro do sensor referência os valores são satisfatórios.

4.7.4 Sensor de ruído

O sensor de ruído escolhido, como não é um dispositivo muito sofisticado, apenas faz uma medida da contagem ADC correspondente à pressão de som que recebe no diafragma. Ou seja, não tem um sinal de saída digital nem está bem calibrado para medições precisas do nível de pressão sonora em dB ou dB(A). Não será possível, neste caso, determinar valores dB(A), diretamente a partir do sensor disponível, pois não existem condições experimentais para tal. Ainda se estudou a possibilidade de aplicar um filtro ponderado A mas esta opção seria bastante complexa a nível de eletrónica e exigia recursos não compatíveis com os objetivos deste trabalho. Além disso, a capacidade de processamento do microcontrolador do Arduino Uno é limitada, o que inviabiliza esta solução.

Portanto, tendo em conta o material disponível, tentou-se estudar o sensor para verificar qual a sua capacidade de resposta em relação àquilo que se quer medir (ruído essencialmente de conversação), e encontrar a melhor solução para atingir este objetivo. Assim, como não existe uma folha de produto para este sensor integrado nem informação disponível (o fabricante não facultou as características do produto), foi necessário realizar um conjunto de testes para caracterizar o sensor quando ligado ao Arduino Uno e qual a sua capacidade de resposta comparando com um sensor de som calibrado.

Teste 1

Foi então efetuado um primeiro teste, de modo a avaliar a resposta em frequência do microfone Electret. O Teste 1 foi efetuado com fontes de som bem conhecidas como diapasões (virtuais), nomeadamente os de frequências 169 Hz, 329,63 Hz, 440 Hz, 523,25 Hz. Para este teste realizaram-se medições com uma taxa de amostragem de 425 valores a cada 50 milissegundos (máximo conseguido devido às limitações do Arduino Uno) para cada diapasão, em condições de silêncio do ambiente circundante, onde se verificava apenas ruído residual,. Os resultados foram bastante satisfatórios pois, através do cálculo das frequências pela análise dos gráficos resultantes, estas verificaram-se bastante aproximadas às reais, com um erro médio inferior a 2%. Como se pode verificar na Fig. 31, o sensor consegue reproduzir toda a onda acústica gerada pelo diapasão de 440 Hz, o mesmo acontecendo com os outros referidos.

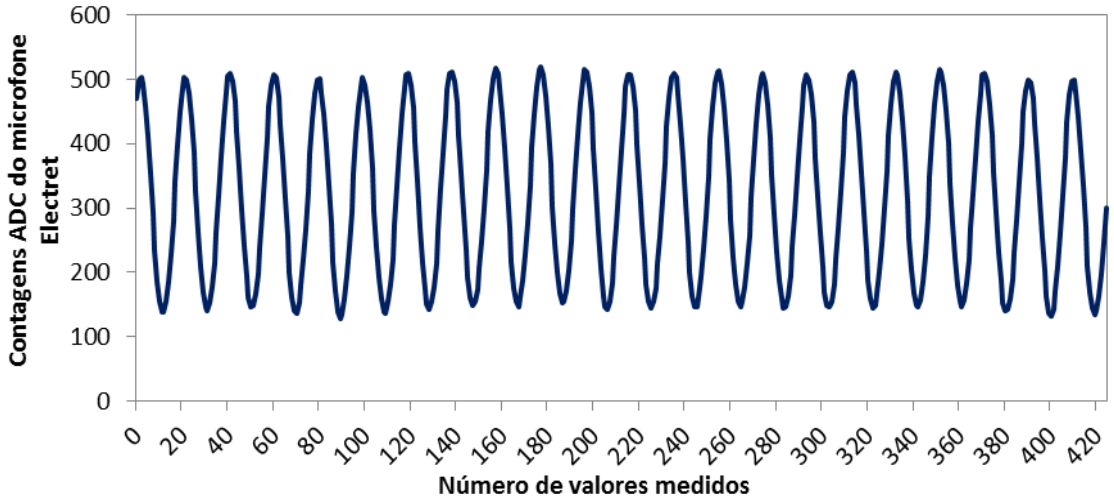

Fig. 31 – Gráfico das medições efetuadas pelo sensor de ruído (contagens ADC) para o diapasão de 440 Hz

Teste 2

Um segundo teste foi posto em prática, desta vez com a medição de uma onda composta pelos diapasões de frequências 196 Hz, 329,63 Hz e 440 Hz, nas mesmas condições de amostragem e de medição. Neste caso foi feito um processamento dos dados, através da aplicação do algoritmo da Transformada Rápida de Fourier (FFT), para decompor a onda nas diferentes frequências. O resultado permitiu verificar que o sensor mais uma vez conseguiu fazer a medição completa das três frequências, captando a onda composta na sua totalidade (ver Fig. 32).

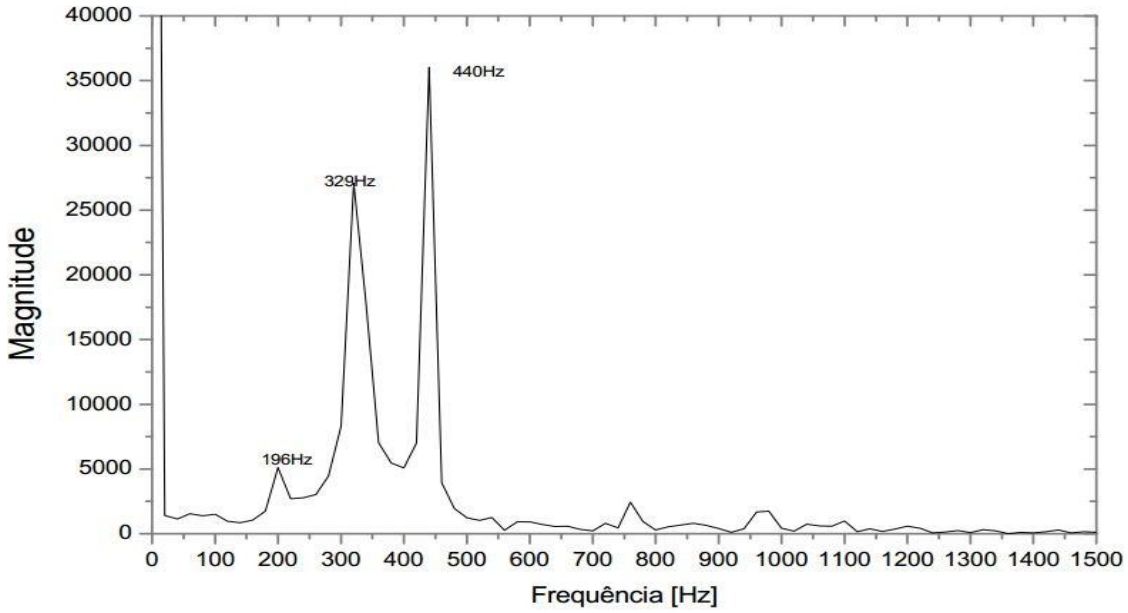

Fig. 32 - Gráfico da decomposição da onda medida pelo sensor de ruído nas três frêquencias da onda composta (FFT)

Teste 3

Com o objectivo de estudar o comportamento do microfone para diferentes valores de pressão sonora realizou-se um teste em que a distância entre a fonte e o microfone varia. Realizou-se então, para o Teste 3, uma experiência com uma duração total de 18 minutos. Em primeiro lugar, o microfone Electret foi colocado durante 2 minutos dentro de uma caixa com isolamento sonoro, para poder verificar qual o mínimo medido.

Posteriormente o sensor de ruído foi colocado a medir uma fonte de som proveniente de duas colunas de computador comuns, às distância de 1 cm, 2 cm, 4 cm, 8 cm, 16 cm, 32 cm e 64 cm. O teste foi realizado durante 2 minutos para cada distância, através da medição de uma onda de som gerada por um diapasão virtual de frequência 440 Hz, com um volume de som médio das colunas. Para 1 cm foi ainda efetuada mais uma medição, também de 2 minutos, com o mesmo diapasão mas com o máximo de volume de som conseguido pelas colunas, de modo a tentar encontrar a zona de saturação do sensor.

Neste teste cada medição teve a duração de aproximadamente 1 segundo com uma amostragem de 1024 valores, das quais é calculado o valor RMS. Foram efetuadas 6 medições por minuto, ou seja, um total de 12 valores RMS para cada distância referida. Através do tratamento dos dados foi traçado

um gráfico com as médias dos valores RMS das contagens ADC a cada 2 minutos. Como resultado obteve-se a curva apresentada no gráfico da Fig. 33 obtida pelos pontos correspondentes aos valores das médias. Tal como seria de esperar a pressão sonora respeita uma lei na qual existe uma relação de proporcionalidade inversa com a distância (*1/r*).

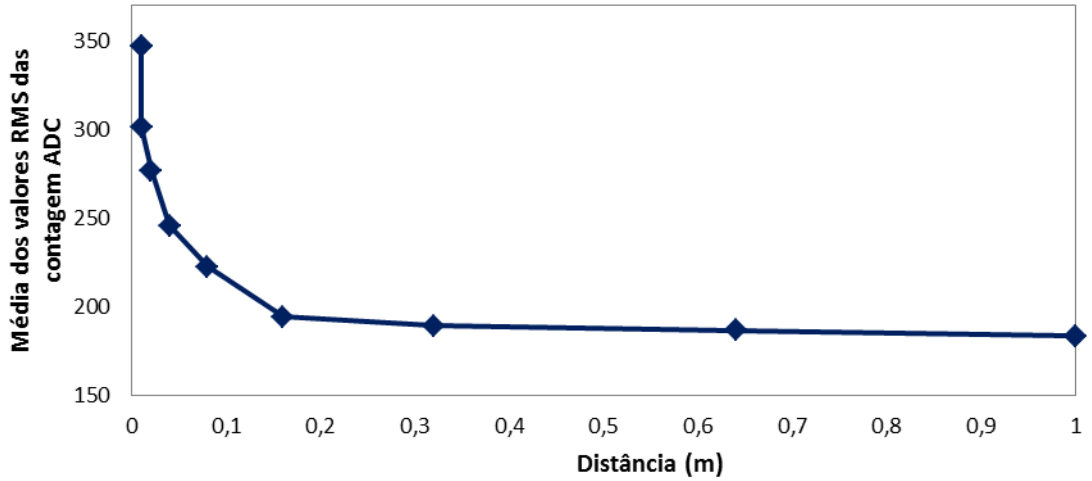

Fig. 33 – Gráfico das médias das medições efetuadas no Teste 3 para as diferentes distâncias e valores máximos e mínimos de som

Para verificar se os resultados correspondem ao comportamento descrito, procedeu-se à linearização dos dados obtidos e com isso traçou-se a reta de tendência (ver Fig. 34). Assim, através do coeficiente de correlação da reta que é 0,89, pode aferir-se que os resultados são bastante satisfatórios pois este valor é muito aproximado do valor ótimo (R^2=1) .

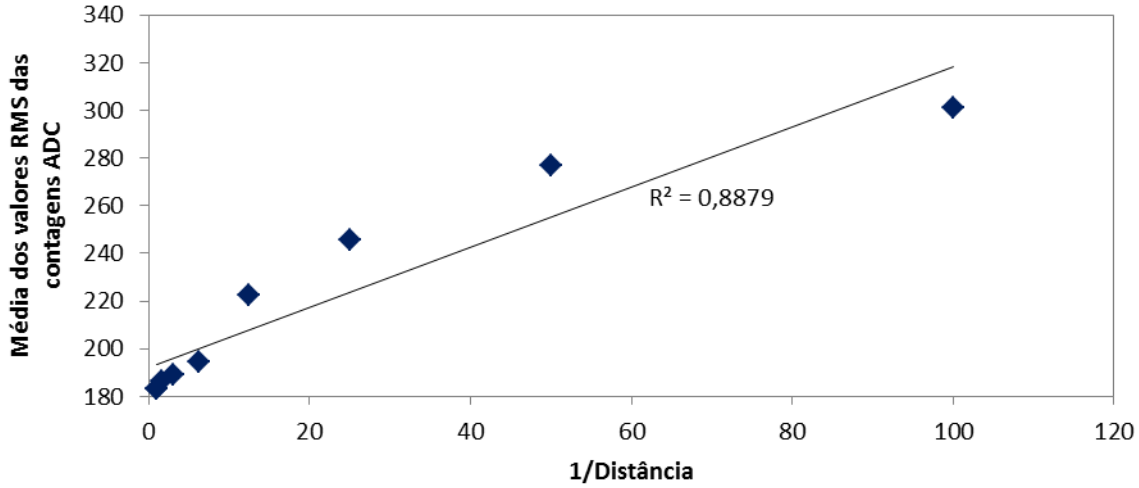

Fig. 34 - Gráfico da média dos valores RMS das contagem ADC em função do inverso da distância

Teste 4

Neste 4º teste, procedeu-se, então, a uma calibração (por aproximação) do Eletret microfone através da comparação da resposta obtida por este com a resposta do sensor calibrado Sound Level Meter 407764, produzido pela EXTECH Instruments. Para isto, realizou-se o Teste 4 onde se usaram, mais uma vez, umas colunas normais de computador a emitir o som de um gerador (virtual) de ruído branco. Colocaram-se os 2 sensores à mesma distância da fonte (1 m) e variou-se o volume do som numa escala de 0 a 20, definida pelo potenciómetro das colunas. Para esta calibração foi utilizado o ruído branco, pois este caracteriza-se por gerar um sinal aleatório que combina, em simultâneo, sons de todas as frequências e tem uma densidade espetral regular em todas elas. Na realidade o intervalo de frequências, neste caso, é determinado pelo aparato experimental pois é influenciado pelas colunas e computador que acabam por filtrar determinadas frequências. Apesar de este não caracterizar a voz humana, é, dentro dos recursos disponíveis, aquele que mais se aproxima daquilo que se pretendia para este teste (avaliar a resposta do sensor em todas as frequências).

Ao contrário dos ensaios anteriores em que o microfone Electret foi testado isoladamente, o Teste 4 foi efetuado já com o sistema todo montado, juntamente com todos os outros sensores, pois é nestas condições que se quer saber qual a sua resposta. A amostragem utilizada foi similar à do Teste 3, no entanto são apenas realizadas 5 medições por minuto (devido à sobrecarga de processamento de dados dos restantes sensores, este é o máximo conseguido pelo Arduino). Começando com silêncio (dentro das condições experimentais possíveis), a cada minuto incrementou-se uma unidade no volume de som e registaram-se os valores obtidos para os dois sensores, microfone Electret e Sound Level Meter. Através desta experiência foram obtidas as curvas de resposta apresentadas na Fig. 35.

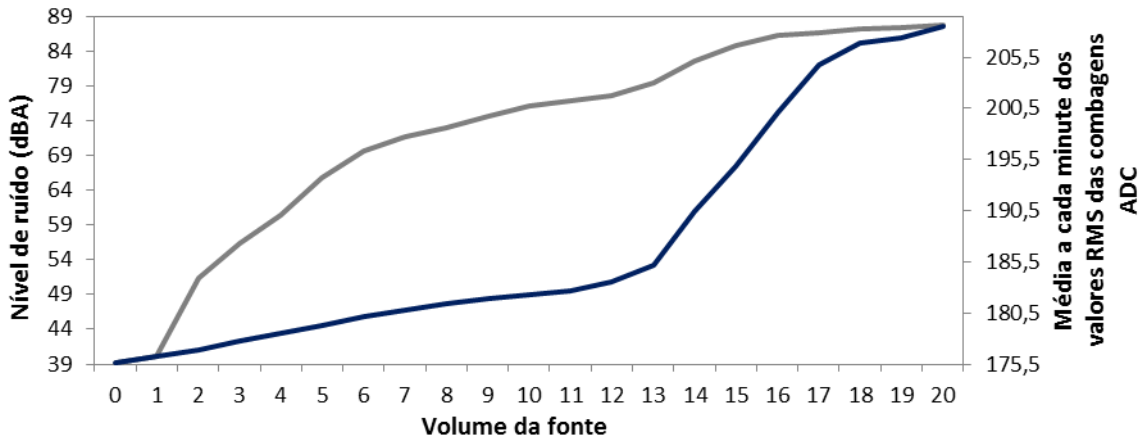

Fig. 35 – Curvas de resposta do Sound Level Meter (linha a azul) e do microfone Electret (linha a cinzento) referentes ao Teste 4

Pela observação da Fig. 35 é possível perceber que os dois sensores têm curvas de resposta distintas, no entanto, estes dados constituem um bom ponto de partida para se poder fazer uma comparação entre os dois em termos de valores medidos. Posto isto, e tendo em conta os testes anteriores, optou-se como solução fazer uma medição do ruído com um objetivo mais qualitativo do que quantitativo, em que se procura estabelecer uma relação entre os valores obtidos (no Teste 4) pelo sensor utilizado neste trabalho e valores dB(A) medidos pelo sensor calibrado (ver Fig. 36).

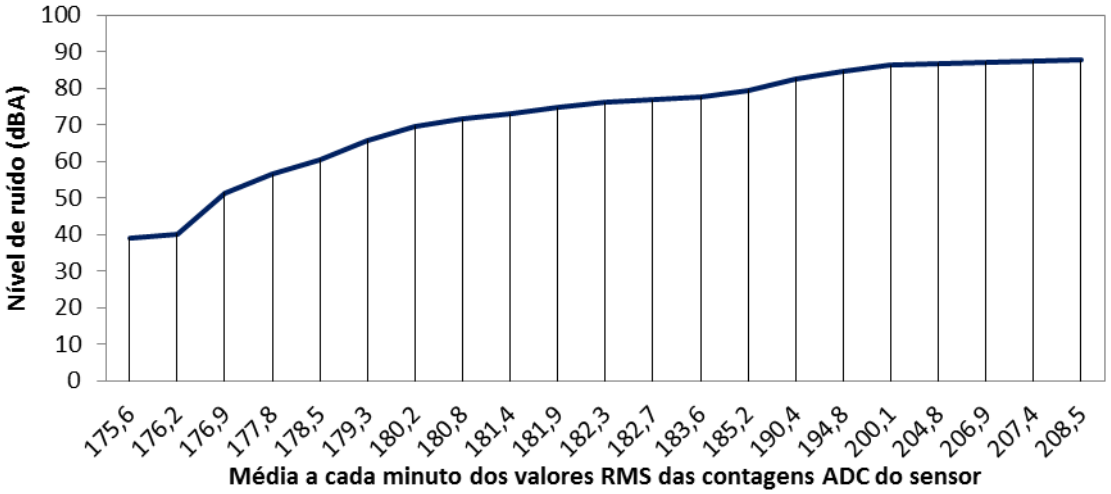

Fig. 36 – Gráfico dos valores registados pelo Sound Level Meter em função dos valores registados pelo Microfone Electret durante o Teste 4

De forma a estabelecer esta relação de correspondência dos valores dos sensores, dividiu-se a curva em várias retas, delimitadas pelas linhas verticais do gráfico da Fig. 36 e determinaram-se as equações de cada reta. Assim é possível, através da equação de cada secção da curva, calcular e atribuir um valor de dB(A) para cada valor medido pelo microfone Electret. Note-se mais uma vez que esta é uma aproximação com algumas limitações e que deve ser utilizada apenas para fins qualitativos na medição de ruído.

Na tabela do Anexo B são apresentadas as equações das retas referidas, através das quais se pode fazer o cálculo do nível de ruído em dB(A). Para determinar este valor é necessário substituir x pelo valor obtido em cada medição (por minuto) do microfone Electret.

Teste 5

Um 5º e último teste foi efetuado e consistiu na comparação dos valores obtidos a partir das equações determinadas para o microfone Electret com os valores medidos pelo Sound Level Meter. Este teste foi efetuado no mesmo modelo de medições do teste 4, que acabou por ser o modelo final utilizado nas medições efetuadas com o microfone Electret.

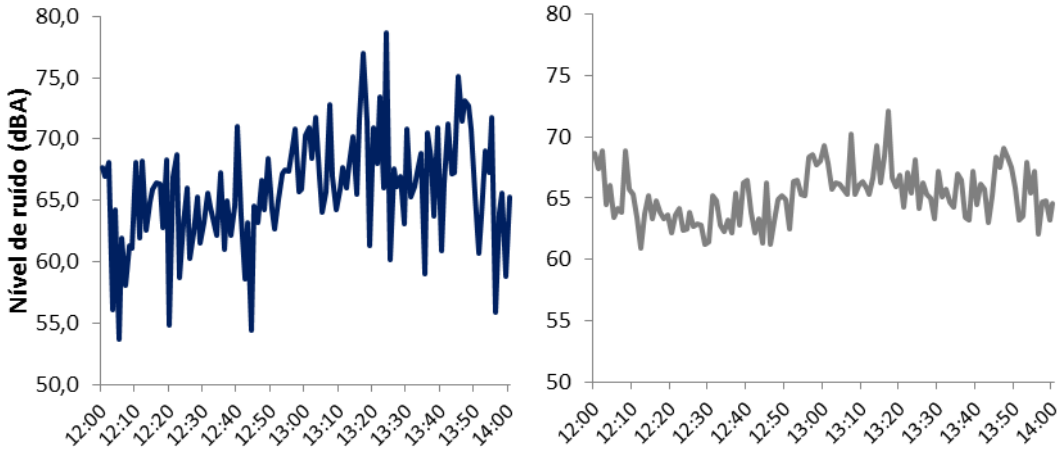

Fig. 37 - Valores obtidos pelo microfone Electret à esquerda e valores medidos pelo Sound Level Meter à direita

Verifica-se pela Fig. 37 que apesar da semelhança entre as curvas dos valores obtidos, o gráfico do microfone Electret parece um pouco ampliado em relação ao Sound Level Meter. Isto porque os picos superiores e inferiores têm valores mais elevados e reduzidos, respetivamente, quando comparados com os picos do gráfico do sensor calibrado. No entanto, as oscilações das curvas de ambos os sensores coincidem, sendo que o desvio médio entre os valores é de ±3%. O erro do sensor referência é de 1,5dB, no entanto não foi possível determinar o erro do microfone Electret. Pode assim considerar-se que as medições do sensor usado para este trabalho são razoáveis e com um desvio aceitável, quando comparado com um sensor calibrado, pois ambos medem na mesma ordem de grandeza.

5. Aplicação do sistema proposto

Após a construção de todo o sistema, incluindo software desenvolvido e estrutura física, foi efetuado um período de medições de forma a demonstrar que este funciona, recolhe dados contínuos e de qualidade. Estas medições, por questões de tempo, foram realizadas apenas num espaço do tipo café/restaurante, mais precisamente no bar do edifício C1 na Faculdade de Ciências da Universidade de Lisboa.

O bar do C1 é um estabelecimento com aproximadamente 220 m^2 onde se servem pequenos-almoços, almoços e lanches. O seu horário de funcionamento durante o período de medições está apresentado na Tabela 10. Como período de ocupação, para efeitos de análise e de cálculos, foi considerado o tempo decorrido desde a chegada das funcionárias até à sua saída.

Tabela 10 - Horário de funcionamento do bar do C1

Dia	**Horário**	
27, 30 e 31 de Julho	7h30	Chegada das funcionárias
	8h – 19h	Horário de expediente
	19h15	Saída das empregadas
1 e 2 de Agosto	8h00	Chegada das funcionárias
	8h30 – 18h	Horário de expediente
	18h15	Saída das empregadas
28 e 29 de Julho	Encerrado ao fim de semana	

Para climatização o bar tem um sistema de ar-condicionado que é ligado de manhã, quando chegam as funcionárias do estabelecimento, e desliga automaticamente a seguir à hora de fecho do mesmo. Este sistema está programado para climatizar o espaço a 25ºC. No estabelecimento encontram-se várias máquinas frigoríficas e outros equipamentos de cozinha e existe também uma televisão que funciona durante todo o horário do expediente. A ventilação do espaço é feita por ventiladores de extração.

Este foi o local escolhido, pois tendo em conta a data em que foi possível iniciar as medições, ou seja, fora da época de aulas, é um dos únicos locais na faculdade onde se verifica circulação de pessoas, principalmente nas horas de almoço e lanche. Assim considerou-se este local como o mais apropriado, tendo em conta que apresenta um perfil de ocupação com rotinas diárias mais ou menos definidas. O fluxo de pessoas neste espaço obedece a um padrão que é determinado pelas horas das refeições acima mencionadas. O facto de possuir rotinas conhecidas é um ponto importante para uma posterior análise dos parâmetros em estudo.

No presente trabalho foi efetuado um período de medições curto, com a duração de 7 dias, deste o dia 27 de Julho às 0h até dia 2 de Agosto às 23h59. Por questões de logística a localização das medições não pôde ser feita no centro do espaço, sendo por isso efetuada junto a uma parede do bar.

A recolha de dados foi feita de forma contínua durantes os 7 dias, com a exceção de duas lacunas que existem nos dados por falha do sistema são estas das 12h02 às 12h09 do dia 30 de Julho e das 12h27 às 12h31 do dia 21 de Julho. O número de intervalos definido para frequência de medições foi de 1 minuto, pois decidiu-se que os resultados seriam suficientemente representativos com esta taxa de recolha de dados. No que se refere ao sensor de ruído este faz 5 medições por minuto e a cada minuto

reporta a média dos valores RMS dessas medições. Sendo este valor transformado em valores dB(A) através das equações determinadas no teste de calibração número 4.

Assim, com os valores obtidos nas medições foi possível traçar alguns gráficos e verificar a evolução de cada parâmetro estudado ao longo do tempo. De seguida serão apresentados e analisados os gráficos obtidos a partir das medições efetuadas no espaço referido, em função da hora e data fornecidos pelo relógio de tempo real incorporado no sistema. Os dados foram também tratados em Excel, o que tornou possível construir gráficos com a distribuição temporal das categorias de conforto definidas pela norma EN 15251. Estes gráficos evidenciam a contribuição individual de cada categoria durante o tempo de ocupação no período das medições.

5.1 Medições e Resultados

5.1.1 Temperatura do ar

A evolução temporal dos valores de temperatura do ar registados, no espaço monitorizado durante toda a semana de medições, é apresentada na Fig. 38.

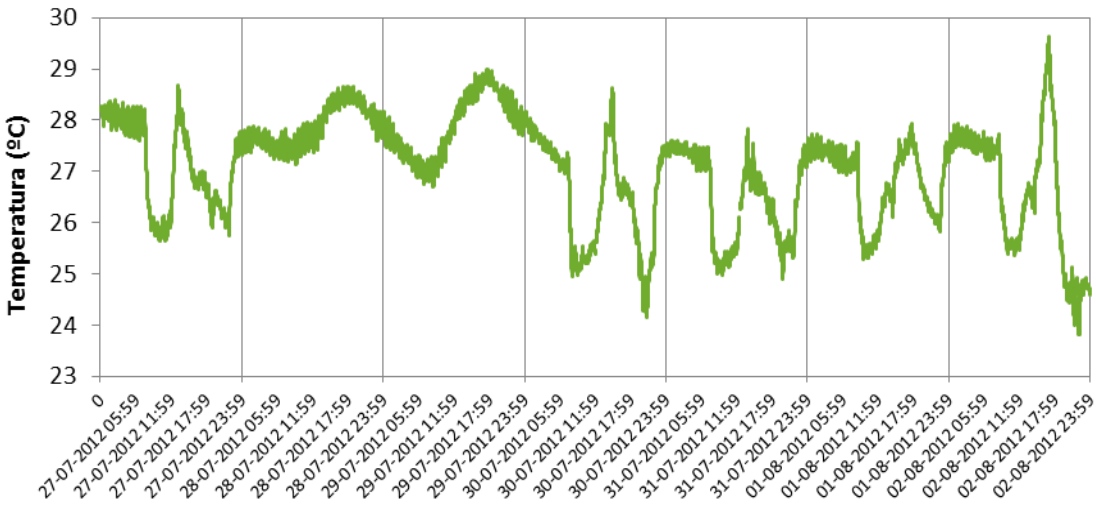

Fig. 38 – Gráfico dos valores de temperatura do ar registados no bar do C1 durante o período de medições (fim de semana nos dias 28 e 29)

Pela observação do gráfico da Fig. 27 pode verificar-se que existem algumas variações nos valores registados para a temperatura no bar do C1 ao longo de cada dia, no entanto esta permanece sempre entre os 24ºC e os 30ºC, sensivelmente. Estas variações são diferentes quando se comparam os dias úteis e os dias não úteis.

Nos dias úteis, os valores mínimos registados verificam-se ao fim da tarde quando o espaço apresenta a ocupação mínima e o ar-condicionado ainda está em funcionamento (exceto nos dias 27 de Julho e 1 de Agosto), enquanto os valores máximos são registados no período de almoço (ocupação máxima). Note-se que nos dias 31 de Julho à tarde e 2 de Agosto à tarde a temperatura desce quase aos 24ºC, provavelmente porque se configurou o sistema de climatização para essa temperatura.

No que se refere aos dias 28 e 29, ou seja, durante o fim de semana, o comportamento da curva da temperatura é diferente dos restantes dias, apresentando uma evolução em que se verifica a descida dos valores durante a noite, onde atingem os valores mínimos, e uma subida durante o dia, onde se registam os valores máximos. Estes valores devem-se ao facto de o bar estar fechado neste período,

logo sem ocupação nem climatização, e portanto acompanha as variações de temperatura exteriores (não foi possível obter a série temporal das temperaturas exteriores).

Para melhor analisar o que acontece durante um dia útil no período de ocupação do espaço traçou-se o gráfico da Fig. 39.

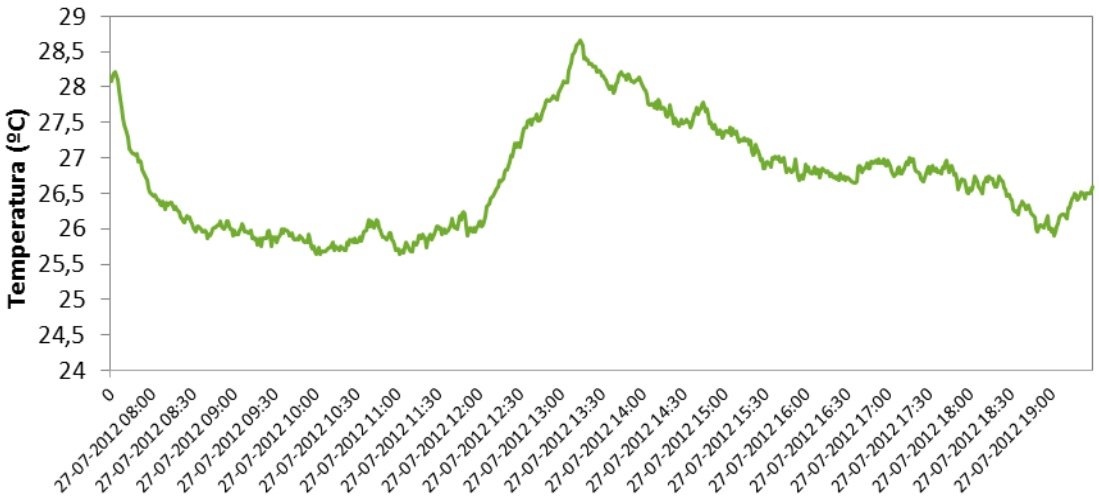

Fig. 39 - Gráfico dos valores de temperatura para um dia útil durante o horário de ocupação do espaço (7h30 às 19h15)

Pela observação do gráfico da figura acima é possível notar que, a partir 7h30 da manhã, existe uma redução significativa da temperatura do bar e um aumento da mesma a seguir às 19 horas. Isto pode ser explicado pelo horário de funcionamento do estabelecimento, sendo que a descida de temperatura se deve à chegada das funcionárias, que ligam o sistema de ar-condicionado, climatizando o espaço. No período de almoço, aproximadamente das 12 às 14 horas, os valores voltam a subir, desta vez pela maior circulação de pessoas e maior número de ocupantes, que aqui se deslocam para almoçar e também maior utilização dos equipamentos, contribuindo para o aquecimento do espaço devido ao calor libertado pelo organismo dos mesmos, equipamentos e comida. O mesmo acontece, mas com uma menor amplitude, por volta das 15h30 até às 17h30 pois é hora do lanche e volta a existir uma maior ocupação.

O aumento da temperatura a seguir às 19 horas é causado pela interrupção do funcionamento do ar-condicionado quando o bar é encerrado, não havendo climatização do espaço, sendo que a temperatura interior passa a ser influenciada somente pela temperatura exterior ajustando-se progressivamente a esta, devido às trocas de calor existentes.

Após o tratamento dos dados procedeu-se à classificação das condições deste parâmetro no bar do C1 segundo os métodos da norma 15251. Na Fig. 40 é possível verificar a percentagem de tempo que o espaço esteve em cada uma das categorias estipuladas na norma referida, tendo por base os valores retirados da mesma.

Através do gráfico da Fig. 40 observa-se que, durante o tempo de ocupação do espaço, no que concerne à temperatura do ar, o bar do C1esteve 53% do tempo dentro das categorias consideradas aceitáveis para satisfazer o conforto térmico dos ocupantes (categorias I,II e III) e 47% do tempo fora dos limites recomendados de temperatura para se estar numa condição de conforto térmico, ou seja, classificado na categoria IV.

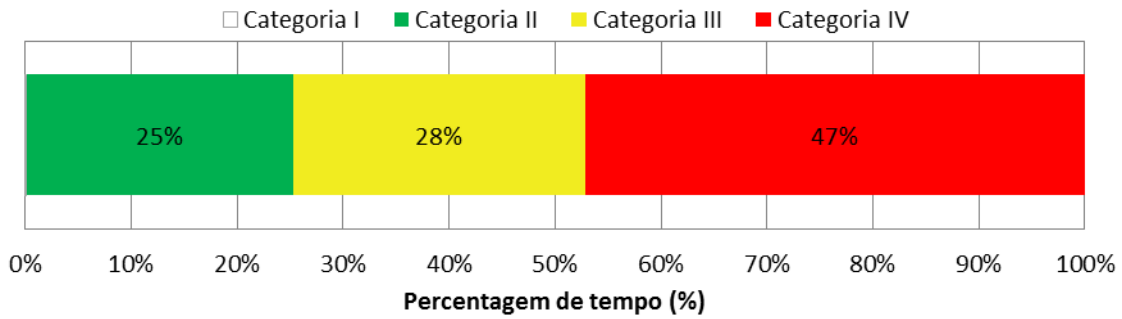

Fig. 40 - Distribuição das categorias para a temperatura do ar durante os períodos de ocupação no total das medições

5.1.2 Humidade relativa

No gráfico presente na Fig. 41 são apresentados os valores medidos para as condições de humidade relativa durante o período total de medições.

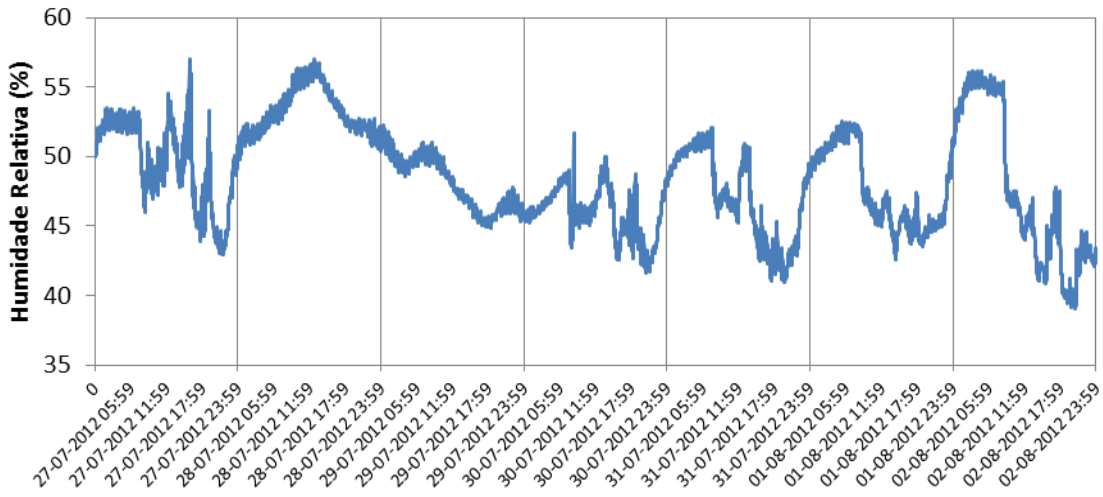

Fig. 41 – Gráfico dos valores de humidade relativa registados no bar do C1 durante o período de medições (fim de semana nos dias 28 e 29)

Através da observação do gráfico da figura acima apresentada, verifica-se que ao longo de cada dia da semana de medições existem variações na humidade relativa no interior do bar do C1, sendo que esta varia entre os 40% e os 60%, não ultrapassando estes limites.

Excetuando a manhã de sábado dia 28 de Julho em que a humidade relativa aumentou, durante quase todo o fim de semana existe uma redução contínua dos valores de humidade relativa que se deve, provavelmente, ao facto de o espaço estar desocupado durante esse período e portanto não existir nenhuma fonte de vapor de água, como, por exemplo, a respiração humana ou as tarefas de cozinha.

Para uma melhor observação de um dia útil no horário de ocupação traçou-se um gráfico do dia 27 de Julho durante este período (ver Fig. 42).

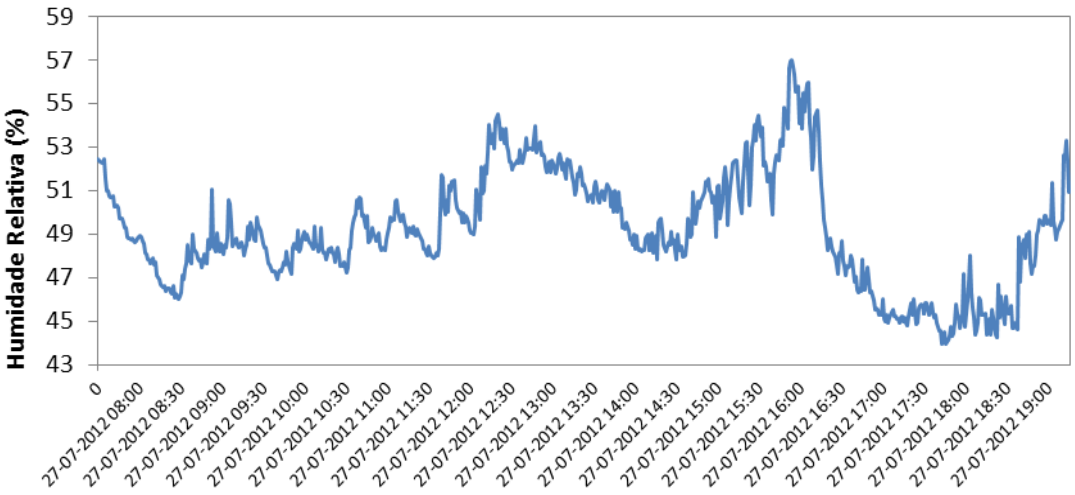

Fig. 42 - Gráfico dos valores de humidade relativa para um dia útil durante o horário de ocupação do espaço (7h30 às 19h15)

A observação do gráfico da figura acima permite verificar que a humidade relativa é influenciada pelo funcionamento do sistema de climatização do espaço, reduzindo-a. Pois, tal como acontece com a temperatura, durante a manhã, logo a seguir a ser ligado o ar-condicionado, nota-se uma descida dos valores deste parâmetro e o contrário acontece após a hora de fecho do estabelecimento, quando este é desligado. Durante a hora de almoço e na hora do lanche existem dois picos onde a humidade relativa atinge os valores máximos, isto devido ao aumento do número de ocupantes no espaço e às tarefas de cozinha que contribuem para a produção de vapor de água.

No gráfico da Fig. 43 é apresentada a percentagem de tempo em que o espaço monitorizado se encontra em cada uma das categorias de classificação.

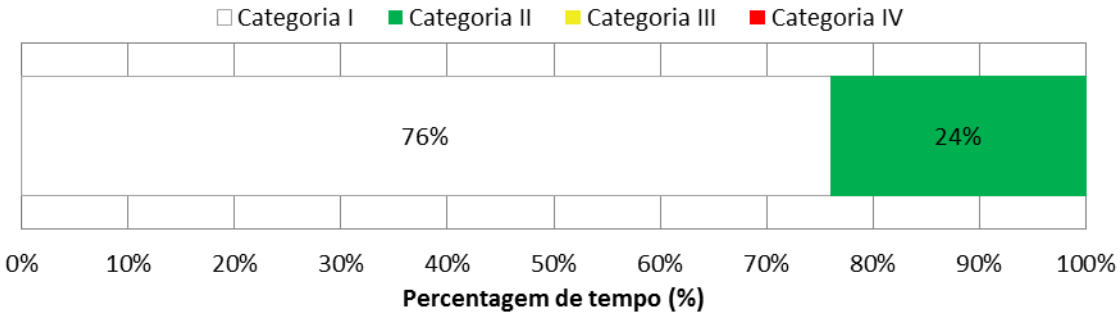

Fig. 43 - Distribuição das categorias para a humidade relativa durante o período de ocupação no total das medições

Assim, o gráfico acima mostra que para este parâmetro o espaço interior monitorizado esteve durante todo o tempo de ocupação dentro das categorias com expectativa mais exigente, pois por 76% da percentagem de tempo classifica-se na categoria I e 24% do tempo na categoria II.

5.1.3 Concentração de CO_2

Através dos valores registados durante a semana de medições foi possível traçar o gráfico da evolução da concentração de CO_2 no interior do bar do C1 (ver Fig. 44).

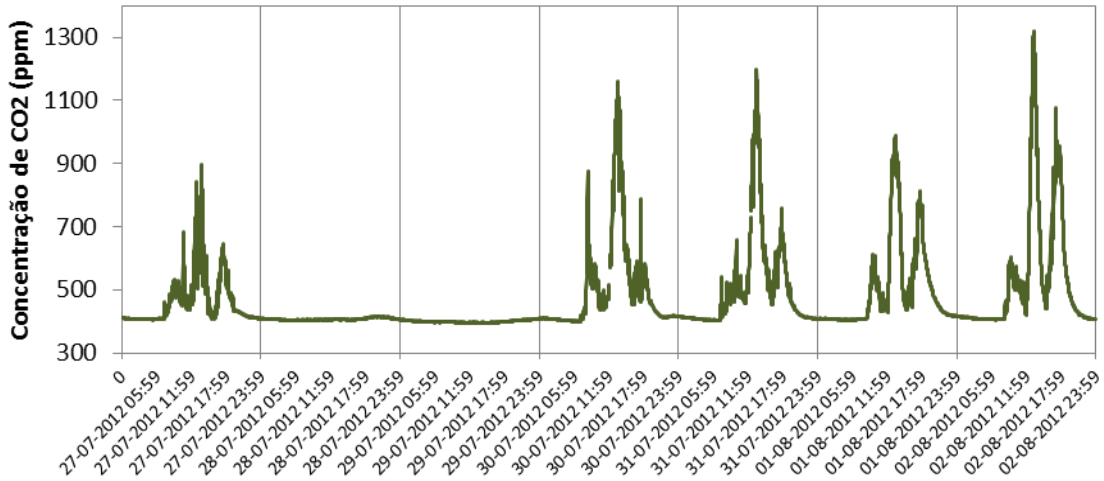

Fig. 44 – Gráfico dos valores de concentração de CO_2 registados no bar do C1 durante o período de medições (fim de semana nos dias 28 e 29)

Os resultados apresentados na figura acima mostram que a presença de CO_2 no espaço depende fortemente da presença humana, aumentando significativamente com o número de ocupantes. Como se pode observar, durante a noite e ao fim de semana (28 e 29 de Julho), quando o espaço está desocupado, a concentração deste gás permanece estável em valores na ordem dos 400 ppm, que representa o valor mínimo registado durante toda a semana, e só apresenta variações durante os períodos em que há ocupação. O valor máximo registado aconteceu no dia 2 de Agosto e é aproximadamente 1300 ppm.

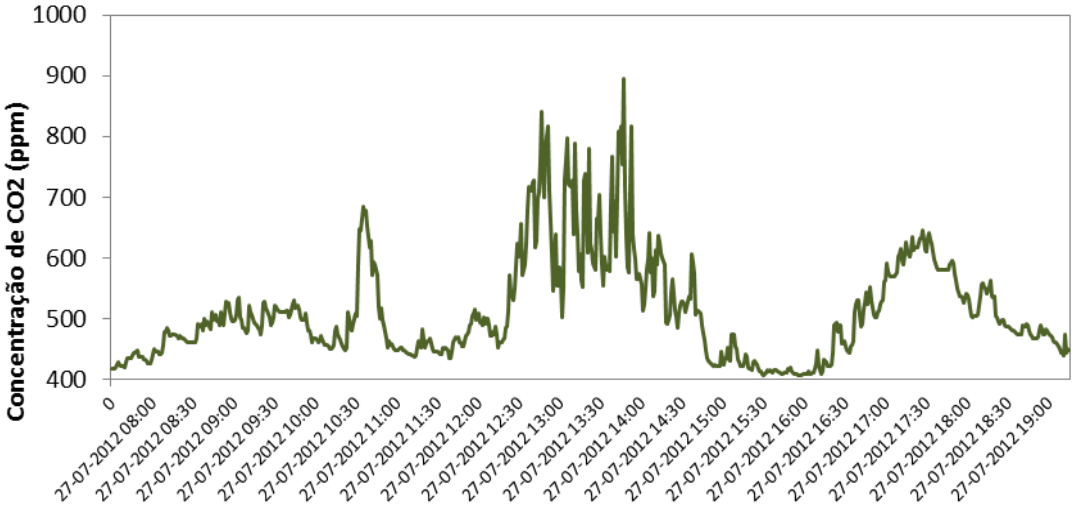

Fig. 45 - Gráfico dos valores de concentração de CO_2 para um dia útil durante o horário de ocupação do espaço (7h30 às 19h15)

Pela observação da Fig. 45 é possível ver com maior detalhe quais as alterações registadas na concentração de CO_2, durante um dia útil no horário de ocupação do espaço. Pode verificar-se, então, que logo que as funcionárias chegam ao estabelecimento a concentração de CO_2 começa a subir apresentando um pico aproximadamente às 10h30 e a variação mais expressiva encontra-se das 12h às 14h, sendo que das 16h às 17h30 volta a registar novo pico. Estes picos são explicados pelo aumento do número de ocupantes no espaço durante os períodos referidos, pois estes coincidem com as horas de pequeno-almoço, almoço e lanche respetivamente. Visto que este parâmetro aumenta com o

número de ocupantes, era de esperar que na hora de almoço se verificasse o valor máximo, o que realmente acontece, sendo este aproximadamente 900 ppm.

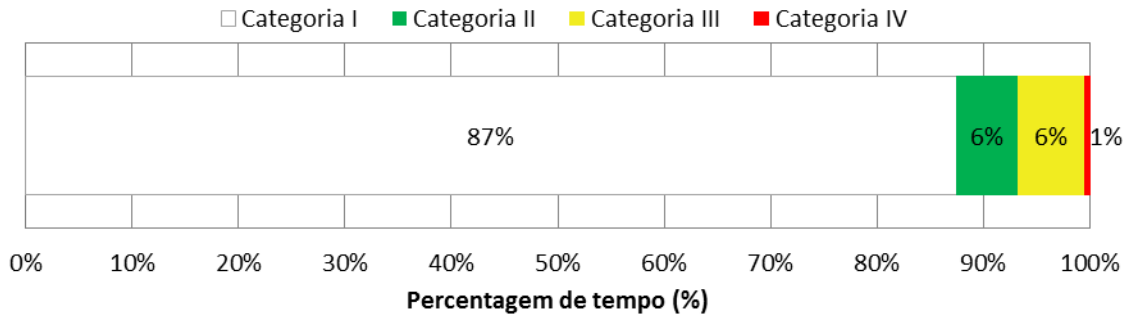

Fig. 46 - Distribuição das categorias para a concentração de CO_2 durante o período de ocupação no total das medições

O gráfico da Fig. 46 apresenta a distribuição temporal das categorias em relação à concentração de CO_2. Através deste verifica-se que durante o período de ocupação o espaço pode ser na maior parte do tempo (87%) classificado na categoria I. No que diz respeito às outras categorias, o espaço esteve 6% do tempo com um ambiente classificável como categoria II, o mesmo acontece para a categoria III e apenas em 1% do tempo este pode ser classificado na categoria IV.

5.1.4 Iluminância

Na Fig. 47 pode ser observado o gráfico traçado com os valores de iluminância medidos pelo ALVI durante a semana de medições.

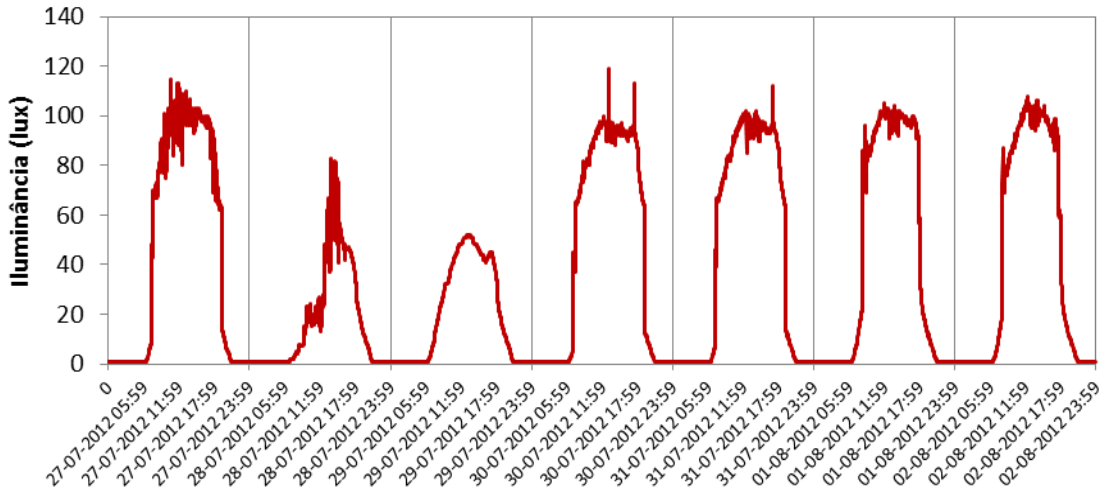

Fig. 47 – Gráfico dos valores de iluminância registados no bar do C1 durante o período de medições (fim de semana nos dias 28 e 29)

O gráfico (ver Fig. 47) mostra a evolução que este parâmetro de conforto visual regista ao longo de cada dia. As variações existentes obedecem ao horário de funcionamento do bar e não às rotinas diárias de fluxo de pessoas, ao contrário do verificado para os restantes parâmetros. Nota-se que os

valores de iluminância acompanham o nível de luminosidade exterior ao longo de cada dia coincidindo os máximos registados (na ordem dos 100/120 lux) com as horas de maior luz natural e os mínimos (0 lux) com o período da noite, sendo também influenciados pela iluminação artificial durante o horário de funcionamento. Estes valores de iluminância medidos são um pouco mais baixos do que se esperava, isto explica-se com a localização das medições ser a um canto onde não existe muita luminosidade, sendo que no centro do espaço se verificam valores na ordem dos 300lux.

É possível verificar que durante os dias de fim de semana o valores máximos atingidos são menores do que nos dias úteis. O que se deve ao facto de não ser ligada a iluminação artificial pois o bar do C1 está encerrado. Este resultado significa que a iluminação artificial contribui significativamente para a iluminação do espaço, já que, de uma forma geral, os valores apresentados nos dias não úteis (só iluminação natural) são aproximadamente metade daqueles que se registam nos dias úteis (iluminação natural + iluminação artificial). São verificados estes valores pois os estores não se encontram completamente abertos.

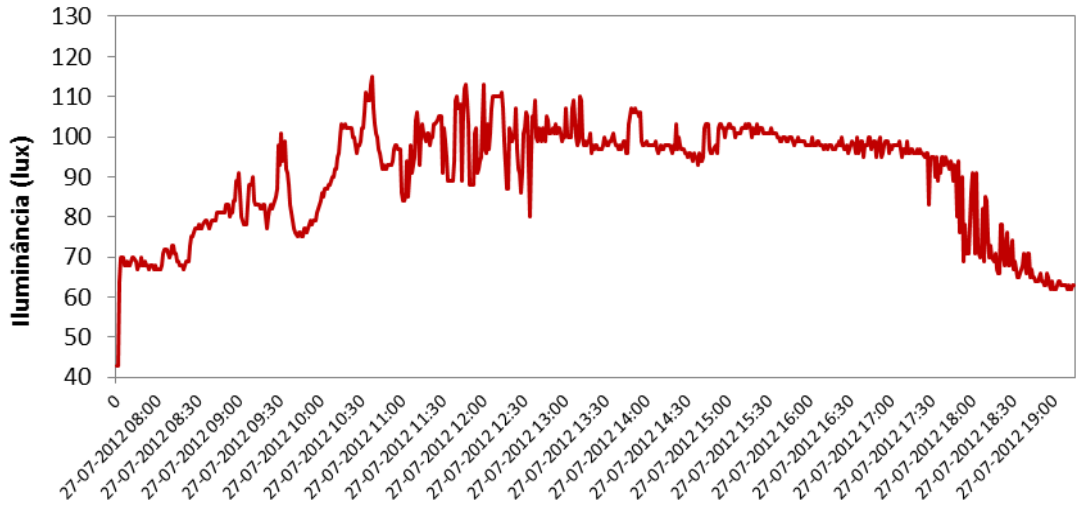

Fig. 48 - Gráfico dos valores de iluminância para um dia útil durante o horário de ocupação do espaço (7h30 às 19h15)

O gráfico acima (Fig. 48), apesar de apresentar maior detalhe, não possibilita retirar muito mais ilações além daquelas observadas no gráfico do tempo total das medições. Verifica-se, portanto, que tal como no gráfico da Fig. 47 os valores de iluminância são maioritariamente influenciados pelas horas de sol e pelo iluminação artificial, não havendo alterações provocadas pelo aumento da ocupação nas horas das refeições, o que seria de esperar. No entanto, nota-se que de manhã, às 7h30, com a chegada das funcionárias há um aumento brusco da iluminância, que pode ser explicado pelo facto de ser ligada a iluminação artificial.

Pode verificar-se uma grande variação dos valores de iluminância aproximadamente à hora de almoço, provavelmente devido à passagem ou presença de pessoas que provocam o sombreamento do sensor.

No que diz respeito a este parâmetro, e para um espaço deste tipo não existe classificação através das categorias estipuladas na norma EN 15251, nem valores recomendados para a iluminância, portanto não é possível construir um gráfico com a percentagem de tempo em que o espaço esteve em condições de conforto ou desconforto.

5.1.5 Nível de ruído

A Fig. 49, que se segue, mostra os resultados das medições efetuadas para o nível de ruído no espaço monitorizado.

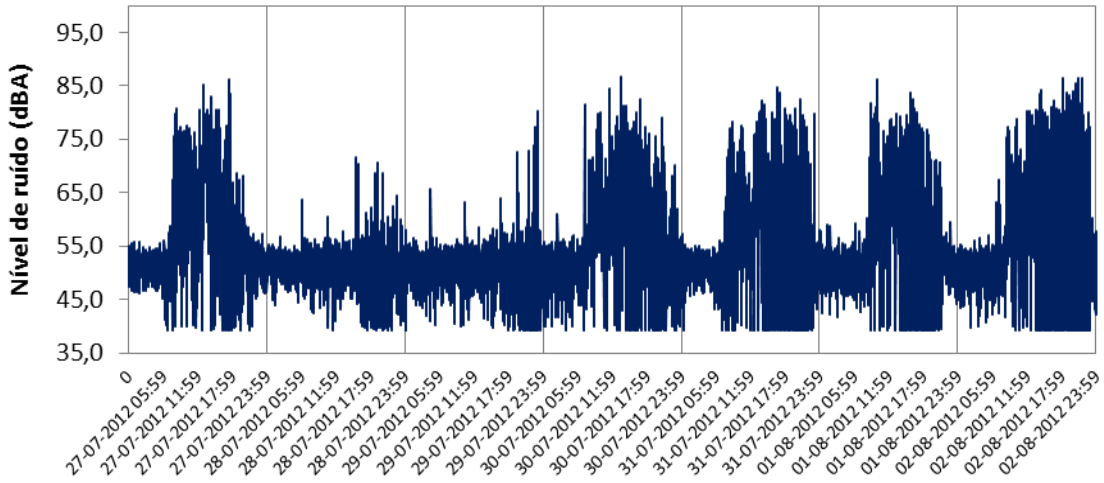

Fig. 49 – Gráfico dos valores do nível de ruído registados no bar do C1 durante o período de medições (fim de semana nos dias 28 e 29)

O gráfico da figura acima mostra que existe uma grande variação do nível de ruído em todos os dias úteis, principalmente nas horas de ocupação do espaço, que vai desde os 39 até aos 85 dB(A), sensivelmente. Enquanto que nos dias não úteis, essa oscilação é mais suave, sendo que atinge os mesmos valores mínimos enquanto que os valores máximos apresentam menor densidade no gráfico e são na ordem dos 70 dB(A).

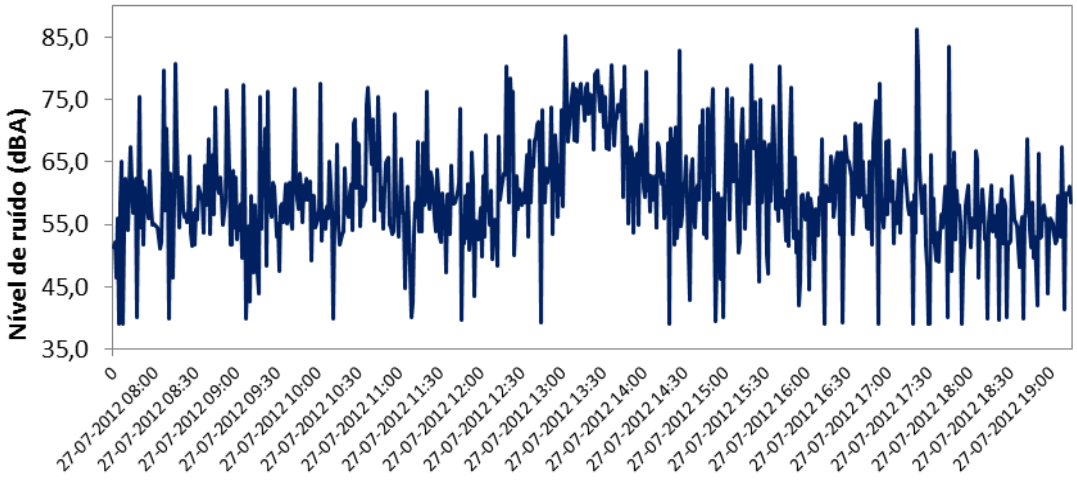

Fig. 50 - Gráfico dos valores do nível de ruído para um dia útil durante o horário de ocupação do espaço (7h30 às 19h15)

O gráfico da Fig. 50 mostra as medições de um dia útil, onde se podem ver com mais pormenor as variações existentes durante o período de ocupação do espaço. O gráfico apresenta uma grande frequência de oscilações, sendo difícil distinguir algum padrão na evolução deste parâmetro. No entanto, é possível verificar que no período de almoço os valores são de uma forma global mais elevados, o que se deve à maior ocupação. Nas restantes horas ao longo do dia também se atingem valores elevados, mas aparentemente sem um padrão definido. Estes podem ser explicados tendo em conta o ruído produzido pelos equipamentos presentes no espaço, bem como pela televisão existente, que está sempre a funcionar com o volume relativamente alto.

Este parâmetro não tem definidos os valores recomendados por categoria, no entanto decidiu-se fazer um gráfico (ver Fig. 51) que mostra a percentagem de tempo em que, para este parâmetro, o bar do C1 esteve em condições de conforto ou desconforto. Considera-se então como conforto os valores menores ou iguais a 45 dB(A) e desconforto valores superiores a este limite.

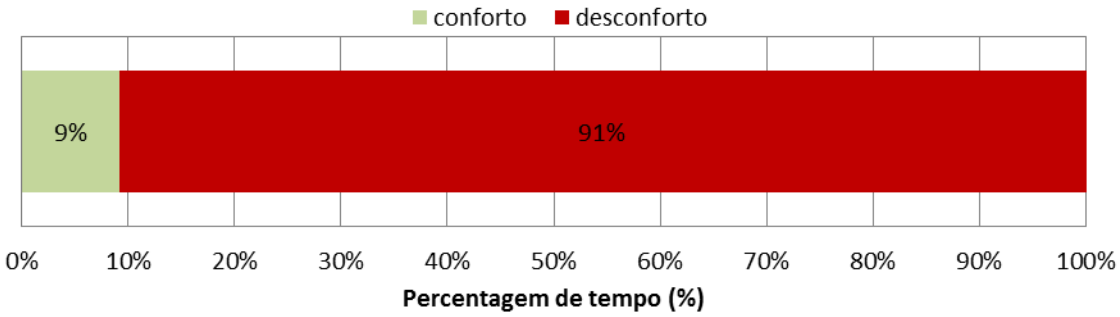

Fig. 51 – Distribuição temporal das condições de conforto e desconforto acústico durante o período de ocupação no total das medições

O gráfico acima evidencia o facto de que, durante a maior parte do tempo de ocupação, o bar apresenta condições de desconforto acústico, registando níveis de ruído acima do recomendado. Como pode verificar-se, apenas em 9% do tempo o estabelecimento apresenta um ambiente acústico considerado confortável.

6. Conclusões

Tendo em conta os objetivos e caraterísticas propostas para o projeto e construção do protótipo, pode dizer-se que o sistema de monitorização remota de condições de conforto interior em edifícios foi concluído com êxito, pois todos os pressupostos iniciais foram alcançados. O ALVI é capaz de fazer uma monitorização contínua de todos os parâmetros de conforto para três tipologias diferentes de espaços, ao mesmo tempo que avalia a qualidade do ambiente interior de acordo com os valores recomendados e categorias de exigência estabelecidas na norma EN 15251. Efetua transferência de dados remotamente através de uma rede sem fios, armazena informação localmente, é portátil, disponibiliza a informação num suporte de dados online e tem uma interface própria para visualização dos dados em tempo real.

Apesar do sucesso, foram encontradas algumas dificuldades durante o desenvolvimento do sistema que acabaram por ser solucionadas. Destacam-se os contratempos na integração do módulo Wifly com o Arduino Uno, pois a comunicação entre estes dois dispositivos é complexa e difícil de estabelecer. Salienta-se também todo o processo de caracterização e calibração do sensor de ruído, que não dispunha de informação suficiente e é um dispositivo limitado para as funções pretendidas neste trabalho. A este propósito deve referir-se que o ruído é um parâmetro difícil de estudar e que requer uma eletrónica mais complexa.

No que diz respeito aos problemas de funcionamento do sistema pode referir-se que este apresenta falhas na ligação à rede sem fios. Estas devem-se à própria robustez da rede Sensor e/ou a problemas no router, que por vezes interrompe a emissão da rede. Isto pode ser solucionado introduzindo uma antena emissora que possibilite um sinal de rede mais forte e também pela substituição do router utilizado.

O ensaio de calibração do sistema realizado para validação dos dados medidos pelo mesmo foi bem-sucedido. Levando em consideração os erros de medição, quase todos os sensores utilizados no protótipo desenvolvido produziram valores bastante aproximados aos obtidos pelos dispositivos referência, ou seja, com uma percentagem de desvio baixa, concluindo-se por isso que estes recolhem dados satisfatórios. Os desvios mais elevados nos resultados do sensor SHT15 em relação à humidade relativa e do Senseair K30 em relação ao IAQ-Calc 7545 podem ter sido influenciados pela necessidade de atualizar a calibração deste último.

Após esta calibração, efetuou-se um período de medições com a duração de uma semana num dos espaços do campus, onde foram recolhidos dados de todos os parâmetros de conforto. Isto provou que o sistema consegue fazer medições de forma contínua durante vários dias. Procedeu-se ao tratamento dos dados medidos e traçaram-se alguns gráficos para posterior análise. Conclui-se, então, que a presença humana no espaço afeta significativamente todos os parâmetros medidos, contribuindo para o seu aumento, nomeadamente no que diz respeito à temperatura do ar, humidade relativa, concentração de CO_2 e nível de ruído. Em relação à iluminância, também é afetada por este fator mas de forma indireta pois não sofre influência do fluxo de pessoas no espaço mas sim do horário de expediente do mesmo, quando a iluminação artificial é ligada. É provável que estes parâmetros também sejam afetados pelo funcionamento dos equipamentos que se encontram dentro do estabelecimento e que estão sempre ligados, contudo não é possível verificar em que medida isto acontece (por estarem sempre a funcionar não causam variações percetíveis). Apenas no caso dos aparelhos que estão ligados só no horário do expediente se podem ver alterações, nomeadamente o aumento do nível de ruído causado pela televisão e pelo funcionamento do ar-condicionado bem como a redução da temperatura do ar e da humidade relativa também por ação do equipamento de ar-condicionado.
A construção dos gráficos que mostram a distribuição temporal das categorias de classificação da QCI permitiu também aferir sobre as condições do ambiente interior, durante o tempo de ocupação do espaço monitorizado. Assim, para a temperatura do ar, verifica-se que o espaço se encontra apenas

metade do tempo numa condição de conforto, o que significa que a outra metade está fora dos limites recomendados, não se considerando portanto um bom ambiente térmico. O desconforto causado por este parâmetro está sempre associado a valores elevados de temperatura e nunca a valores baixos, uma vez que durante o período de medições esta nunca atinge valores menores que 23ºC, como seria de esperar na época do ano em que foram efetuadas as medições.
Para a variável humidade relativa, o espaço encontra-se na totalidade do tempo classificado nas categorias I e II, portanto está, nesta época do ano, sempre em condições de conforto. No que diz respeito à concentração de CO_2, o estabelecimento encontra-se na zona de desconforto em apenas 1% do tempo de ocupação, concluindo-se que de um modo geral este apresenta uma boa qualidade do ar.
No que se refere à iluminância do espaço, como não existem valores mínimos ou máximos recomendados, considera-se que este está sempre em condições de conforto visual durante o tempo de ocupação. Em relação ao nível de ruído, pode dizer-se que este é o parâmetro mais problemático dentro do espaço estudado, pois 91% do tempo é passado numa condição de desconforto acústico. Todavia, deve ser tido em conta mais uma vez que os valores de dB(A) produzidos pelo sensor utilizado são uma aproximação, podendo não corresponder exatamente à realidade.
Num cômputo geral, pode então concluir-se que os critérios de conforto onde se deve intervir de modo a aumentar a satisfação dos ocupantes são a temperatura do ar, o nível de ruído e talvez a concentração de CO_2.

A realização deste trabalho foi bastante enriquecedora pois permitiu adquirir novos conhecimentos e competências em áreas diversificadas como a eletrónica, programação, comunicação e redes, assim como trabalhos de oficina (soldadura e montagem do sistema).

O sistema proposto pode ainda ser utilizado como ferramenta para o estudo da produtividade dos ocupantes em função das condições do ambiente em que se inserem, bem como para determinar e sugerir medidas de melhoria de conforto e/ou de eficiência energética.

7. Referências bibliográficas

Alfano, Francesca R. d'Ambrosio, et al. 2010. *Indoor Environment and Energy Efficiency in Schools.* Brussels : REHVA, 2010. ISBN 978-2-930521-03-9.

Alves, Jorge Maia e da Graça, Guilherme Carrilho. 2011. Desenvolvimento de um sistema portátil para avaliação de condições de conforto interior em edifícios. *Proposta de colaboração do SESUL.* Lisboa : s.n., 2011.

Arduino Team. 2012. *Arduino.* [Online] ©Arduino , 2012. http://www.arduino.cc/.

Arezes, Pedro Miguel. 2002. Percepção do Risco de Exposição Ocupacional ao Ruído. s.l. : Escola de Engenharia da Universidade do Minho, 2002.

ASHRAE. 2004. *ASHRAE Standard 55 "Thermal Environmental Conditions for Human Occupancy".* Atlanta : American Society of Heating, Refrigerating and Air Conditioning, 2004.

ASHRAE.2009. THERMAL COMFORT. *ASHRAE Handbook - Fundamentals (SI).* 2009.

Banzi, Massimo. 2011. *Getting Started with Arduino.* U.S.A : O'Reilly Media, Inc., 2011. SBN: 978-1-449-309879 .

Bronsema, Ben e Björck, Maria. 2004. *Performance Criteria of Buildings for Health and Comfort.* s.l. : ISIAQ-CIB Task Group, 2004.

Brüel & Kjær. 1993. Sound Intensity. 1993.

CO2Meter.com. 2010. Datasheet: K-30 Sensor. 2010. Ficha do produto.

Concha-Barrientos, Marisol, Campbell-Lendrum, Diarmid e Steenland, Kyle. 2004. Occupational noise. *Assessing the burden of disease from work-related hearing* . s.l. : World Health Organization, 2004. Vol. 9, Environmental Burden of Disease Series. ISBN 92 4 159192 7.

Corgnati, Stefano Paolo, da Silva, Manuel Gameiro e Ansaldi, Roberta. 2011. *Indoor Climate Quality Assessment.* Finland, Forssa : REHVA, 2011. ISBN 978-2-930521-05-3.

Doukas, Charalampos. 2012. *Building Internet of Things With the Arduino.* Charleston : s.n., 2012. Vol. I. 1470023431.

Emmerich, Steven J. e Persily, Andrew K. 2001. State-of-the-Art Review of CO2 Demand Controlled Ventilation Technology and Application. s.l. : National Institute of Standards and Technology, 2001. NISTIR 6729.

EN 12464-1. 2002. Norma Europeia EN 12464-1. *Light and lighting - Lighting of work places - Part 1: Indoor.* 2002. Bruxelles. EN 12464-1:2002.

EN 15251. 2007. European Standard EN 15251. *Indoor environmental input parameters for design and assessment of energy performance of buildings addressing indoor air quality, thermal environment, lighting and acoustics.* s.l., Brussels, Belgium : International Organization for Standardization, 2007. EN 15251:2007:E.

Fraden, Jacob. 2010. *Handbook of Modern Sensors.* New York : Springer, 2010. ISBN 978-1-4419-6465-6.

2012. Github. *harlequin-tech / WiFlyHQ.* [Online] GitHub Inc., 2012. [Citação: 15 de 06 de 2012.] https://github.com/.

Groth, Andreas. 2007. ENERGY EFFICIENCY BUILDING DESIGN GUIDELINES FOR BOTSWANA. *Section 4 - INDOOR ENVIRONMENT.* Botswana : s.n., 2007. Ministry of Minerals, Energy and Water Resources.

Hamamatsu. 2010. Photodiode Technical Information. 2010. http://sales.hamamatsu.com/en/support/technical-notes.php.

ISO 7730. International Standard 7730 (2005). 3 *Ergonomics of the thermal environment — Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria.* Geneva, Switzerland : International Organization for Standardization. ISO 7730:2005:E.

Jung, Walt. 2005. *Op Amp Apllications Handbook.* Oxford : Analog Devices, Inc., 2005. ISBN 0-7506-7844-5.

Karvinen, Kimmo e Karvinen, Tero. 2011. *Make: Arduino Bots and Gadgets.* Canada : O'Reilly Media, Inc, 2011. 978-1-449-38971-0.

Khedari, Joseph, et al. 2000. Thailand ventilation comfort chart. 2000. Energy and Buildings. 32 2000 245–249.

Knowles Acoustics. 2004. PRODUCT SPECIFICATION. 2004. Vol. A, Ficha do produto. MD9745APZ-F.

Louçano, Nelson Ramos. 2009. Eficiência energética em edifícios: Gestão do sistema iluminação. Bragança : s.n., 2009.

Margolis, Michael. 2011. *Arduino Cookbook.* United States of America : O'Reilly Media, Inc., 2011. Vol. II. 978-0-596-80247-9.

Metrel. Multinorm MI 6201 User Manual. 20 751 260 .

Networks, Roving. 2011. RN-XV Data Sheet. 2011. Vol. v0.3, Ficha do produto.

Pais, Aida e Bettencourt, Rui. Condições de iluminação em ambiente de escritório: .

PCB Piezotronics, Inc. Microphone Handbook. *Test and Measurement Microphones.*

Prill, Rich. 2000. Why Measure Carbon Dioxide Inside Buildings? Havana : Washington State University Extension Energy Program, 2000. WSUEEP07-003.

Reas, Casey e Fry, Ben. 2009. *Interview with Casey Reas and Ben Fry*. 23 de Setembro de 2009. http://rhizome.org/editorial/2009/sep/23/interview-with-casey-reas-and-ben-fry/.

Semicondutor, Dallas. 2009. DS1307. 2009. Ficha do produto.

Senseair. 2000. Theory of Calibration for SenseAir IR gas sensors . *Technical Note.* 2000. TN-010.

Sensirion. 2011. Datasheet SHT1x (SHT10, SHT11, SHT15). *Humidity and Temperature Sensor IC.* Switzerland : s.n., 2011. Vol. v.05, Ficha do produto.

Silvestre, Ana Rita. 2011. Desenvolvimento de um Método Simplificado para Cálculo de Caudal de Ar Novo a partir de Medições de Concentração de CO2 em Edifícios de Serviços. 2011. Tese de Mestrado.

SparkFun Electronics. 2012. *Sparkfun Electronics.* [Online] 2012. http://www.sparkfun.com/.

Taos Inc. 2005. TSL2560, TSL2561 LIGHT TO DIGITAL CONVERTER. 2005. Ficha do produto. TAOS059D.

Telecom Regulatory Authority. 2003. Wifi Technology. 2003. Technical Affairs & Technology Sector.

TSI Incorporated. 2009. NDIR CO2 Sensing Technology. *Application Note TSI-037.* USA : TSI Incorporated, 2009.

TSI Instruments Ltd. 2007. Certificate of Calibration and Testing Model 7545. 2007.

Zumtobel Staff. 2004. *The Lighting Handbook.* 2004.

8. Anexo A: Preços do Material

Componente	**Preço s/ IVA (€)**	**Distribuidor**
Arduino Uno Ver. 3	20	Inmotion
Wireless SD Shield	19,9	Inmotion
Wifly RN-XV	29,95	Inmotion
Sensor de temperatura e humidade SHT15	32,95	Inmotion
Sensor de CO_2 Senseair K30	53	CO2meter
Sensor de intensidade luminosa TSL2561	11,18	watterott
Sensor de ruído Microfone Electret	5,9	PTRobotics
Relógio de tempo real RTC	9,9	PTRobotics
Material diverso	10	vários
TOTAL	192,78	

9. Anexo B: Tabela com as equações das retas que são utilizadas para o cálculo do nível de ruído por comparação dos sensores

Volume	Sensor Calibrado [dB(A)]	Microfone Electret (Média dos valores RMS calculados)	Equação da reta
0 - 1	39,1 - 40,2	175,6 - 176,2	$y = 1{,}83 * x - 282{,}83$
1 - 2	40,2 - 51,3	176,2 - 176,9	$y = 15{,}86 * x - 2753{,}83$
2 - 3	51,3 - 56,4	176,9 - 177,8	$y = 5{,}67 * x - 951{,}13$
3 - 4	56,4 - 60,5	177,8 - 178,5	$y = 5{,}86 * x - 985{,}00$
4 - 5	60,5 - 65,8	178,5 - 179,3	$y = 6{,}62 * x - 1122{,}06$
5 - 6	65,8 - 69,6	179,3 - 180,2	$y = 4{,}22 * x - 691{,}24$
6 - 7	69,6 - 71,6	180,2 - 180,8	$y = 3{,}33 * x - 531{,}07$
7 - 8	71,6 - 73,0	180,8 - 181,4	$y = 2{,}33 * x - 350{,}27$
8 - 9	73,0 - 74,7	181,4 - 181,9	$y = 3{,}40 * x - 543{,}76$
9 - 10	74,7 - 76,1	181,9 - 182,3	$y = 3{,}50 * x - 561{,}94$
10- 11	76,1 - 76,8	182,3 - 182,7	$y = 1{,}75 * x - 242{,}92$
11 - 12	76,8 - 77,6	182,7 - 183,6	$y = 0{,}89 * x - 85{,}59$
12 -13	77,6 - 79,4	183,6 - 185,2	$y = 1{,}12 * x - 128{,}95$
13 - 14	79,4 - 82,5	185,2 - 190,4	$y = 0{,}60 * x - 31{,}00$
14 - 15	82,5 - 84,7	190,4 - 194,8	$y = 0{,}50 * x - 12{,}70$
15 - 16	84,7 - 86,3	194,8 - 200,1	$y = 0{,}30 * x + 25{,}89$
16 - 17	86,3 - 86,7	200,1 - 204,8	$y = 0{,}08 * x + 69{,}27$
17 - 18	86,7 - 87,2	204,8 - 206,9	$y = 0{,}24 * x + 37{,}94$
18 - 19	87,2 - 87,4	206,9 - 207,4	$y = 0{,}40 * x + 4{,}44$
19 - 20	87,4 - 87,7	207,4 - 208,5	$y = 0{,}27 * x + 30{,}84$

10. Anexo C: Código de programação do Arduino IDE

```
//include libraries
#include <SD.h>
#include <Wire.h>
#include <TSL2561.h>
#include <WiFlyHQ.h>

//Wifly class initiation
WiFly wifly;

//Change these concerning the WiFi network
const char mySSID[] = "Sensor";
const char myPassword[] = "buildingenergy";
const char site[] = "192.168.1.100";

//Define I2C adresses
#define DS1307_ADDRESS 0x68 //set RTC address
int co2Addr = 0x7F; //set CO2 sensor address
TSL2561 tsl(TSL2561_ADDR_FLOAT); //set TLS2561 address

//Temperature sensor variables
int temperatureCommand  = B00000011;  // command used to read temperature
int humidityCommand = B00000101;  // command used to read humidity
const int clockPin = 6;  // pin used for clock
const int dataPin  = 7;  // pin used for data
int ack;  // track acknowledgment for errors
int val;  //value coming from SHT15
float temperature;
float humidity;

//CO2 Sensor variables
float co2Value;
int co2_value = 0;

//Light Sensor variables
int Light;

//Noise sensor variables
const int middleValue = 512; //the middle of the range of analog values
const int numberOfSamples = 1024; //how many readings will be taken each time
float sample_1,sample_2,sample_3,sample_4, sample_5; //the value read from microphone each time
float signal_1,signal_2,signal_3,signal_4, signal_5; //the reading once removed DC offset
float  signalRMS_1,signalRMS_2,signalRMS_3,signalRMS_4,  signalRMS_5;  //the RMS values of that loop of
readings
float signalRMS_average=0;
int i=0;
float Noise;

//SD card Variables
const int CS_pin = 4; //SD card pin
const int pow_pin = 8; //SD card power

//RTC Variables
int seconds;
int minutes;
int hours;
int weekDays;
int monthDays;
int months;
int years;

void setup ()
{
 Serial.begin(9600);// open serial at 9600 bps
 Serial.println("Warm-up system (1 min)");
 wifly.begin(&Serial, NULL);//Start Wifly

 //Join wifi network if not already associated
  if (!wifly.isAssociated()) {
            // Setup the WiFly to connect to a wifi network
            wifly.setSSID(mySSID);
            wifly.setPassphrase(myPassword);

            if (wifly.join()) {
               Serial.println("Joined wifi network");
            } else {
               Serial.println("Failed to join wifi network");
            }
   } else {
      Serial.println("Already joined network");
   }

   wifly.setDeviceID("Wifly-WebClient");

   if (wifly.isConnected()) {
      Serial.println("Old connection active. Closing");
            wifly.close();
   }

   if (wifly.open(site, 1234)) {
      Serial.print("Connected to ");
            Serial.println(site);
    delay (2000);
```

```
            //Send the request
            wifly.println("GET / HTTP/1.0");
            wifly.println();
  } else {
     Serial.println("Failed to connect");
  }

 delay(10000);//Delay used to warm-up the sensores 10sec before start measuring


 //Initialize TSL2561
 Wire.begin ();//Initiate the Wire library and join the I2C bus
 tsl.setGain(TSL2561_GAIN_16X);      // set 16x gain (for dim situations)

 // Changing the integration time gives you a longer time over which to sense light
 tsl.setTiming(TSL2561_INTEGRATIONTIME_402MS);  // longest integration time (dim light)

 //Initialize SD Card
 //CS Pin is an output
 pinMode(CS_pin, OUTPUT);

 //Card will Draw Power from Pin 8, so set it high
 pinMode(pow_pin, OUTPUT);
 digitalWrite(pow_pin, HIGH);

 if (!SD.begin(CS_pin))
 {
   Serial.println("Card Failure");
   return;
 }
}

void loop ()
{
 ReadRTC();

 //Read Noise Sensor
 ReadNoise ();//It has to be outside the next if, otherwise it just measure noise at second zero

 if (seconds == 0) //time when values are saved on micro SD card
 {
  //Read the temperature
  ReadTemperature();
  delay(100);

  //Read the humidity
  ReadHR();
  delay(100);
```

```
 //Read Lux
 ReadLux();
 delay(100);

 //Read CO2 Sensor
 float CO2Value = readCO2()*3.3/5;
 int co2Value = CO2Value;
 delay(100);

 //Save on microSD card
 //The file will be CSV format
 File logFile = SD.open("T1907.csv" , FILE_WRITE);  //Open a file to write to; Only one file can be open at a time
 if (logFile)
 {//print to micro SD card
  logFile.print(monthDays);
  logFile.print("/");
  logFile.print(months);
  logFile.print("/");
  logFile.print(years);
  logFile.print(" ");
  logFile.print(hours);
  logFile.print(":");
  logFile.print(minutes);
  logFile.print(":");
  logFile.print(seconds);
  logFile.print(";");
  logFile.print(temperature);
  logFile.print(";");
  logFile.print(humidity);
  logFile.print(";");
  logFile.print(co2Value);
  logFile.print(";");
  logFile.print(Light);
  logFile.print(";");
  logFile.println (Noise);
  logFile.close();
 }
 else
 {
  Serial.println("Couldn´t open log file");
 }

 //Print by serial to wifly
 printDate();
 wifly.print(temperature);
 wifly.print (";");
 wifly.print(humidity);
```

```
    wifly.print (";");
    wifly.print(co2Value);
    wifly.print (";");
    wifly.print(Light);
    wifly.print (";");
    wifly.println (Noise);
  }
   delay(500);
}

//READ CO2 SENSOR  FUNCTION
int readCO2()
{
 Wire.beginTransmission(co2Addr);
 Wire.write(0x22);
 Wire.write(0x00);
 Wire.write(0x08);
 Wire.write(0x2A);
 Wire.endTransmission ();
 delay (10);
 Wire.requestFrom(co2Addr,4);
 byte i = 0;
 byte buffer[4] = {0,0,0,0};//create an array to input 4 bytes where [0] is the address, [1] is the MSB (Most Significant
Byte),[2] is the LSB (Last Significant Byte) and [3] the cheksum

 while(Wire.available())//while sensor is sending values arduino get them to the array
 {
  buffer[i] = Wire.read();
  i++;
 }
 co2_value = 0;//set the array to 0
 co2_value |= buffer[1] & 0xFF;//reunion (sum) of buffer[1] with an array of 0,0,0,0 that results in buffer[1]
 co2_value = co2_value << 8;//move the value 8 bits back
 co2_value |= buffer[2] & 0xFF;//reunion (sum) of buffer [2] with an array of 0,0,0, that results in buffer[2]

 byte sum = 0;
 sum = buffer[0] + buffer[1] + buffer[2];//Checksum

 if(sum == buffer[3])//verify if the read is complete and right
 {
   return co2_value;  //Success!
 }
 else
 {
   return 0;//Failure
 }
 delay(2000);
}
```

```
//COMMANDS FOR SHT15 SENSOR
// commands for reading/sending data to a SHT15 sensor
int shiftIn(int dataPin, int clockPin, int numBits) {
  int ret = 0;

  for (int i=0; i<numBits; ++i) {
    digitalWrite(clockPin, HIGH);
    ret = ret*2 + digitalRead(dataPin);
    digitalWrite(clockPin, LOW);
  }
  return(ret);
}

// send a command to the SHT15 sensor
void sendCommandSHT(int command, int dataPin, int clockPin) {
  int ack;

  // transmission start
  pinMode(dataPin, OUTPUT);      //configuring pins
  pinMode(clockPin, OUTPUT);

  digitalWrite(dataPin, HIGH);   //transmission sequence
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, LOW);

  // shift out the command (the 3 MSB are address and must be 000, the last 5 bits are the command)
  shiftOut(dataPin, clockPin, MSBFIRST, command);
  //shiftOut - Each bit is written in turn to a data pin, after which a clock pin is pulsed (taken high, then low) to indicate
that the bit is available

  // verify we get the right ACK
  digitalWrite(clockPin, HIGH);   //after the 8th falling edge
  pinMode(dataPin, INPUT);
  ack = digitalRead(dataPin);      //read ACK
  if (ack != LOW)
    Serial.println("ACK error 0");
  digitalWrite(clockPin, LOW);      //SCLK low to continue
  ack = digitalRead(dataPin);
  if (ack != HIGH)
    Serial.println("ACK error 1");  //DATA released and goes high (pull-up)
                                    // it is measuring
}
```

```
// wait for the SHTx answer
void waitForResultSHT(int dataPin)
{
  int ack;

  pinMode(dataPin, INPUT);
  for(int i=0; i<100; ++i)
  {
    delay(10);
    ack = digitalRead(dataPin);
    if (ack == LOW)
      break;                    //SHTx pulls down and enters IdleMode
                              //Value stored until readout
  }
  if (ack == HIGH)
    Serial.println("ACK error 2");  // after waiting time no Idle Mode
}

  // get data from the SHTx sensor
long getData16SHT(int dataPin, int clockPin)
{
  long val;

  // get the MSB (most significant bits)
  pinMode(dataPin, INPUT);
  pinMode(clockPin, OUTPUT);
  val = shiftIn(dataPin, clockPin, 8);  //restar SCLK shiftIn does that
  val *= 256; // this is equivalent to val << 8;

  // send the required ACK
  //for the first Byte
  pinMode(dataPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);

  // get the LSB (less significant bits)
  pinMode(dataPin, INPUT);
  val |= shiftIn(dataPin, clockPin, 8);
  return val;
}

// skip CRC data from the SHTx sensor
void skipCrcSHT(int dataPin, int clockPin)
{
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);    //DATA high
  digitalWrite(clockPin, HIGH);  //SCLK high-low
  digitalWrite(clockPin, LOW);
}

//READ SHT15 FUNCTION (Temperature)
void ReadTemperature()
{
  sendCommandSHT(temperatureCommand, dataPin, clockPin);
  waitForResultSHT(dataPin);
  val = getData16SHT(dataPin, clockPin);
  skipCrcSHT(dataPin, clockPin);
  temperature = (float)val * 0.01 - 40.1;
}

//READ SHT15 FUNCTION (Humidity)
void ReadHR()
{
  sendCommandSHT(humidityCommand, dataPin, clockPin);
  waitForResultSHT(dataPin);
  val = getData16SHT(dataPin, clockPin);
  skipCrcSHT(dataPin, clockPin);
  humidity = -2.0468 + 0.0367 * val + -0.0000015955 * val * val;
}

//READ LUX FUNCTION
void ReadLux()
{
  // Read 'visible' channel (difference between infrared and fullspectrum).
  uint16_t x = tsl.getLuminosity(TSL2561_VISIBLE);

  // Read 32 bits with top 16 bits IR, bottom 16 bits full spectrum
  // That way is possible to do whatever math and comparions
  uint32_t lum = tsl.getFullLuminosity();
  uint16_t ir, full;
  ir = lum >> 16;
  full = lum & 0xFFFF;
  Light = tsl.calculateLux(full, ir);
}

//READ NOISE FUNCTION
void ReadNoise ()
{
  float sumOfSquares_1 = 0;
  float sumOfSquares_2 = 0;
  float sumOfSquares_3 = 0;
  float sumOfSquares_4 = 0;
  float sumOfSquares_5 = 0;
```

```
if(seconds == 0)//1st measure
{
  for (i=0; i<numberOfSamples; i++) //take many readings and average them
  {
    sample_1 = analogRead(0); //take a reading
    signal_1 = (sample_1 - middleValue); //work out its offset from the center
    signal_1 *= signal_1; //square it to make all values positive
    sumOfSquares_1 += signal_1; //add to the total
  }
  signalRMS_1 = sqrt(sumOfSquares_1/numberOfSamples); //calculate RMS value
}

  if(seconds == 12)//2nd measure
{
  for (i=0; i<numberOfSamples; i++) //take many readings and average them
  {
    sample_2 = analogRead(0); //take a reading
    signal_2 = (sample_2 - middleValue); //work out its offset from the center
    signal_2 *= signal_2; //square it to make all values positive
    sumOfSquares_2 += signal_2; //add to the total
  }
  signalRMS_2 = sqrt(sumOfSquares_2/numberOfSamples); //calculate RMS value
}

  if(seconds == 24)//3rd measure
{
  for (i=0; i<numberOfSamples; i++) //take many readings and average them
  {
    sample_3 = analogRead(0); //take a reading
    signal_3 = (sample_3 - middleValue); //work out its offset from the center
    signal_3 *= signal_3; //square it to make all values positive
    sumOfSquares_3 += signal_3; //add to the total
  }
  signalRMS_3 = sqrt(sumOfSquares_3/numberOfSamples); //calculate RMS value
}

  if(seconds == 36)//4th measure
{
  for (i=0; i<numberOfSamples; i++) //take many readings and average them
  {
    sample_4 = analogRead(0); //take a reading
    signal_4 = (sample_4 - middleValue); //work out its offset from the center
    signal_4 *= signal_4; //square it to make all values positive
    sumOfSquares_4 += signal_4; //add to the total
  }
  signalRMS_4 = sqrt(sumOfSquares_4/numberOfSamples); //calculate RMS value
}
```

```
    if(seconds == 48)//5th measure
  {
    for (i=0; i<numberOfSamples; i++) //take many readings and average them
    {
      sample_5 = analogRead(0); //take a reading
      signal_5 = (sample_5 - middleValue); //work out its offset from the center
      signal_5 *= signal_5; //square it to make all values positive
      sumOfSquares_5 += signal_5; //add to the total
    }
    signalRMS_5 = sqrt(sumOfSquares_5/numberOfSamples); //calculate RMS value
  }

  if(seconds == 0)//Average the 6 measures every minute, only 1 minute after system starts
  {
    signalRMS_average=(signalRMS_1 + signalRMS_2 + signalRMS_3 + signalRMS_4 + signalRMS_5)/5;
    Noise=signalRMS_average;
  }
}

//RTC FUNCTIONS
byte bcdToDec(byte val)  {
// Convert binary coded decimal to normal decimal numbers
  return ( (val/16*10) + (val%16) );
}

void ReadRTC()
{
// Reset the register pointer
  Wire.beginTransmission(DS1307_ADDRESS);

  byte zero = 0x00;
  Wire.write(zero);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_ADDRESS, 7);

  seconds = bcdToDec(Wire.read());
  minutes = bcdToDec(Wire.read());
  hours = bcdToDec(Wire.read() & 0b111111); //24 hour time
  weekDays = bcdToDec(Wire.read()); //0-6 -> sunday - Saturday
  monthDays = bcdToDec(Wire.read());
  months = bcdToDec(Wire.read());
  years = bcdToDec(Wire.read());
}

void printDate(){
  //print the date EG   3/1/11 23:59:59
```

```
  wifly.print(monthDays);
  wifly.print(";");
  wifly.print(months);
  wifly.print(";");
  wifly.print(years);
  wifly.print(";");
  wifly.print(hours);
  wifly.print(";");
  wifly.print(minutes);
  wifly.print(";");
  wifly.print(seconds);
  wifly.print(";");
}
```

11. Anexo D: Código de programação do Processing

Anexo D.1: Classe principal do código

```
//Import libraries
import cosm.*;
import napplet.*;//library to use several skeches on one window
import processing.net.*;
Server myServer;

//ESCOLHA DA CATEGORIA E TIPO DE ESPAÇO
/*CATEGORIA DE CLASSIFICAÇÂO DA QUALIDADE DO CLIMA INTERIOR (Norma 15251):
    - CATEGORIA_QCI = 1 -> Categoria I
    - CATEGORIA_QCI = 2 -> Categoria II
    - CATEGORIA_QCI = 3 -> Categoria III
 Tipo de espaço a monitorizar
    - ESPAÇO = A -> Sala de aula
    - ESPAÇO = B -> Escritório
    - ESPAÇO = C -> Café/Restaurante
*/
int CATEGORIA_QCI = 3;
int ESPACO = 'C';

int oldminute;

//Cosm settings
DataOut feed;
String apiKey = "iLr9iaOz-8O3PB704ARunzGApwOSzlDQpQysY3h2UGVmQmlvc2M9";//Each time IP changes is necessary to create a new API Key
String feedId = "70123";//Has to correspond to the ID feed

//variable to set the text file
PrintWriter texto;

//Sensor variables from Arduino Serial port
float yeartime, monthtime, daytime;
float hourtime, minutetime,secondtime;
float oldminutetime=0;
float Temp = 0;
float HR = 0;
float CO2 = 0;
float Lux = 0;
float NoiseL = 0;

//Noise variables
float dBAcalc = 0;
int dBA = 0;
float Noisemin=175.6;
float Noisemax=208.5;

float timeX;

void setup() {
  size(1315, 690);
  background(255);

  //Napplet windows deffinition
  NAppletManager nappletManager = new NAppletManager(this);
  nappletManager.createNApplet("plotsUP", 675, 0);
  nappletManager.createNApplet("Interface_class", 0, 0);
  nappletManager.createNApplet("plotsDOWN", 675, 350);

  //Set server to get values from Sensor Network (Wifly)
  myServer = new Server(this, 1234);

  //Set server to send values to Cosm
  feed = new DataOut(this, apiKey, feedId);  //intantiate feed
  feed.setVerbose(false);  //optional debug info

  //create a text file and whose header
  texto = createWriter("SMQCIdata.csv");
  String header = "Data/Time; Temp (Celsius); HR (%);CO2 (ppm); Light (Lux); Noise (dBA)";
  texto.println(header);
}

void draw() {

  background(255);
  getValuesfromWifly();
  //call calculate_dBA function
  calculate_dBA();

  senddatatoCOSM();

  timeX=60.0*minutetime+secondtime+hourtime*3600.0;//calulate the total number of seconds of 24 hours to map the values to draw plots

  //write values on the text file everytime minute changes
    if(minutetime != oldminutetime )//just print if the time change using second from Arduino Time //print the data received from Arduino on the text file
  {
```

```
    texto.println((int)daytime+"/"+(int)monthtime+"/"+(int)yeartime+"
"+(int)hourtime+":"+(int)minutetime+":"+(int)secondtime + ";" +Temp + ";" + HR + ";" + (int)CO2 + ";" + (int)Lux +
";" + nf(dBA,1,1));

    texto.flush();
  }
  oldminutetime = minutetime;
}

void getValuesfromWifly()
{
// Get the next available client
  Client thisClient = myServer.available();
  // write the output of the client
  if (thisClient !=null) {
    //dataIn = thisClient.read();
    String whatClientSaid = thisClient.readStringUntil('\n');
    if (whatClientSaid != null)
    {
      whatClientSaid = trim(whatClientSaid); // clean the string
      // The data is split into an array of Strings with a comma or asterisk as a delimiter and converted into an array of
integers.
      float [] infos =float (split(whatClientSaid, ";")); //fill in a matrix with the different variables

      if (infos.length >=10) //10 is the number of variables imported
      {
      daytime = infos [0];
      monthtime = infos [1];
      yeartime = infos [2];
      hourtime = infos [3];
      minutetime = infos [4];
      secondtime = infos [5];
      Temp = infos [6];
      HR = infos [7];
      CO2 = infos [8];
      Lux = infos [9];
      NoiseL = infos [10];
      }
    }
  }
}

void senddatatoCOSM() {
  feed.setStream(5, Temp); //send request (datastream id, new value)
  feed.setStream(6, HR); //send request (datastream id, new value)
  feed.setStream(7, CO2); //send request (datastream id, new value)
  feed.setStream(8, Lux); //send request (datastream id, new value)
  feed.setStream(9, dBA); //send request (datastream id, new value)
}

void calculate_dBA ()
{
  //Calculate dBA
 if (NoiseL<Noisemin && NoiseL!=0)
 {
   NoiseL=Noisemin;
 }
 if (NoiseL>Noisemax)
 {
   NoiseL=Noisemax;
 }
 if (NoiseL>=Noisemin && NoiseL<=176.2)
 {
   dBAcalc=1.83333333333335*NoiseL-282.833333333337;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>176.2 && NoiseL<=176.9)
 {
   dBAcalc= 15.8571428571425*NoiseL-2753.8285714285;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>176.9 && NoiseL<=177.8)
 {
   dBAcalc= 5.66666666666663*NoiseL-951.133333333327;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>177.8 && NoiseL<=178.5)
 {
   dBAcalc= 5.85714285714295*NoiseL-985.000000000017;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>178.5 && NoiseL<=179.3)
 {
   dBAcalc= 6.6249999999999*NoiseL-1122.06249999998;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>179.3 && NoiseL<=180.2)
 {
   dBAcalc= 4.22222222222233*NoiseL-691.244444444463;
   dBA=(int)dBAcalc;
 }
 if (NoiseL>180.2 && NoiseL<=180.8)
 {
   dBAcalc= 3.33333333333321*NoiseL-531.066666666644;
   dBA=(int)dBAcalc;
 }
```

```
  if (NoiseL>180.8 && NoiseL<=181.4)
  {
   dBAcalc= 2.33333333333337*NoiseL-350.266666666672;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>181.4 && NoiseL<=181.9)
  {
   dBAcalc= 3.40000000000001*NoiseL-543.760000000001;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>181.9 && NoiseL<=182.3)
  {
   dBAcalc= 3.49999999999993*NoiseL-561.949999999987;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>182.3 && NoiseL<=182.7)
  {
   dBAcalc= 1.75000000000011*NoiseL-242.925000000019;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>182.7 && NoiseL<=183.6)
  {
   dBAcalc= 0.88888888888888*NoiseL-85.5999999999984;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>183.6 && NoiseL<=185.2)
  {
   dBAcalc= 1.12500000000001*NoiseL-128.950000000002;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>185.2 && NoiseL<=190.4)
  {
   dBAcalc= 0.596153846153843*NoiseL-31.0076923076917;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>190.4 && NoiseL<=194.8)
  {
   dBAcalc= 0.5*NoiseL-12.7;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>194.8 && NoiseL<=200.1)
  {
   dBAcalc= 0.30188679245283*NoiseL+25.8924528301887;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>200.1 && NoiseL<=204.8)
  {
   dBAcalc= 0.0851063829787243*NoiseL+69.2702127659573;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>204.8 && NoiseL<=206.9)
  {
   dBAcalc= 0.238095238095239*NoiseL+37.9380952380951;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>206.9 && NoiseL<=207.4)
  {
   dBAcalc= 0.400000000000006*NoiseL+4.43999999999882;
   dBA=(int)dBAcalc;
  }
  if (NoiseL>207.4 && NoiseL<=208.5)
  {
   dBAcalc= 0.272727272727272*NoiseL+30.8363636363639;
   dBA=(int)dBAcalc;
  }
}
```

Anexo D.2: Classe para amostragem dos valores atuais, recomendados e avaliação do QCI

```
public class Interface_class extends NApplet {
int Tmin;
int Tmax;
int HRmin;
int HRmax;
int CO2min;
int CO2max;
int Luxmin;
int Luxmax;
int Noisemin;
int Noisemax;

//Text variables
PFont plotFont;
PFont font;
PFont font1;

//Interface variables
PImage img1;PImage img2;
PImage img3;PImage img4;
PImage img5;PImage img6;
PImage img7;PImage img8;
PImage img9;
```

```
int color1 = #516C7C;
int color2 = 200;
int xe=75;//x ellipse
int ye=75;//y ellipse
int a=115;//y distance among ellipses
int x1=75;//x position of image ellipses
int xgreen=500;//x position of green ellipses
int xyel=xgreen+40;//x position of yellow ellipses
int xred=xyel+40;//x position of red ellipses
int y1=130;//y image ellipse 1
int y2=y1+a;//y image ellipse 2
int y3=y2+a;//y image ellipse 3
int y4=y3+a;//y iamge ellipse 4
int y5=y4+a;//y image ellipse 5

//Text positions
int xtext1= 185;int xtext2= xtext1;int xtext3=xtext1-10;
int xtext4=xtext1-10;int xtext5=xtext1;
int xtext11=340;int xtext12=xtext11;int xtext13=xtext11;
int xtext14=xtext11;int xtext15=xtext11;

void setup ()
{
  size(675,700);
  background(200);
  plotFont = loadFont( "SansSerif.bold-48.vlw" );
  textFont(plotFont);
  smooth();

  //Interface Variables definition
  img1 = createImage(width, height, ARGB);
  img2 = loadImage("termometro.png");
  img3 = loadImage("HR.png");
  img4 = loadImage("co2.png");
  img5 = loadImage("lux.png");
  img6 = loadImage("ruido.png");
  img7 = loadImage("hand.png");
  img8 = loadImage("hand1.png");
  img9 = loadImage("hand2.png");

  for(int i=0; i < img1.pixels.length; i++)
  {
    img1.pixels[i] = color(0, 90, 102, i%img1.width);
  }

  font = loadFont( "SansSerif.bold-48.vlw" );
  font1 = loadFont("Tahoma-Bold-48.vlw");
  textFont(font);
  smooth();
}

void draw ()
{
  //int Temp_int = int (Temp);
  //println(Temp_int);

//INTERFACE DRAWING
  background(200);
  drawImages();
  drawValues();
  drawText();
  drawEllipses();
  IEQ ();
}

//INTERFACE FUNCTIONS
void drawImages()
{
  image(img1, 0, 0);
  //images
  fill(254);
  stroke (1);
  strokeWeight(3);
  ellipse(x1,y1+7,xe,ye);
  image(img2, x1-27, y1-25);
  ellipse(x1,y2+7,xe,ye);
  image(img3, x1-27, y2-25);
  ellipse(x1,y3+7,xe,ye);
  image(img4, x1-25, y3-21);
  ellipse(x1,y4+7,xe,ye);
  image(img5, x1-29, y4-26);
  ellipse(x1,y5+7,xe,ye);
  image(img6, x1-22, y5-26);
  stroke(2);
  smooth();
}

void drawValues ()
{
 //current values
  textFont(font);
  textSize (40);
  if (Temp == 0 && HR == 0 && CO2 == 0 && Lux == 0 && NoiseL ==0)
  {
    text(nf(Temp,1,0),xtext1+10,y1+20); //nf() function sets the number of digits required
```

```
  text(nf(HR,1,0),xtext2+10,y2+20);
  text(nf(CO2,1,0),xtext3+20,y3+20);
  text(nf(Lux,1,0),xtext4+20,y4+20);
  text(dBA,xtext5+10,y5+20);
 }
 else if (Temp > 0 || HR > 0 || CO2 > 0 || Lux > 0 || NoiseL >0)
 {
  text(nf(Temp,1,1),xtext1-20,y1+20);
  text(nf(HR,1,1),xtext2-20,y2+20);
  text(nf(CO2,1,0),xtext3-10,y3+20);
  text(nf(Lux,1,0),xtext4-10,y4+20);
  text(dBA,xtext5-20,y5+20);
 }

 //reference range values
 textFont(font);
 textSize (20);
 if (CATEGORIA_QCI == 1)
 {
 text("21-25",xtext11,y1+20);//Temperature
 text("30-50",xtext12,y2+20);//HR
 text("<730",xtext13,y3+20);//CO2
 text("500",xtext14+10,y4+20);//Lux
 //text("<45",xtext15+18,y5+20);//Noise Level
 }
  if (CATEGORIA_QCI == 2)
 {
 text("20-26",xtext11,y1+20);//Temperature
 text("25-60",xtext12,y2+20);//HR
 text("<880",xtext13,y3+20);//CO2
 text("500",xtext14+10,y4+20);//Lux
 //text("<45",xtext15+18,y5+20);//Noise Level
 }
    if (CATEGORIA_QCI == 3)
 {
 text("19-27",xtext11,y1+20);//Temperature
 text("20-70",xtext12,y2+20);//HR
 text("<1180",xtext13,y3+20);//CO2
 text("500",xtext14+10,y4+20);//Lux
 //text("<45",xtext15+5,y5+20);//Noise Level
 }

  if (ESPACO == 'A' || ESPACO == 'B')
  {
    text("<35",xtext15+5,y5+20);//Noise Level
  }
  if (ESPACO == 'C')
  {
    text("<45",xtext15+5,y5+20);//Noise Level
   }

  //Units
  textSize (15);
  text("ºC",xtext1+80,y1+20);
  text("ºC",xtext11+60,y1+20);
  text("%",xtext2+80,y2+20);
  text("%",xtext12+60,y2+20);
  text("ppm",xtext3+87,y3+20);
  text("ppm",xtext13+60,y3+20);
  text("lux",xtext4+87,y4+20);
  text("lux",xtext14+55,y4+20);
  text("dBA",xtext5+75,y5+20);
  text("dBA",xtext15+50,y5+20);
 }

 void drawText()
 {
  fill(#0C211E);
  textFont (font1);
  textSize (18);
  text("Valor",xtext1,y1-90);
  text("atual",xtext1,y1-70);
  text("Parâmetro",x1-45,y1-90);
  text("monitorizado",x1-55,y1-70);
  text("Qualidade do",485,y1-90);
  text("ambiente interior",470,y1-70);

  if (CATEGORIA_QCI == 1)
  {
    text("Valor ref.",xtext12-10,y1-90);
    text("Categoria I",xtext12-15,y1-70);
  }

    if (CATEGORIA_QCI == 2)
  {
    text("Valor ref.",xtext12-10,y1-90);
    text("Categoria II",xtext12-20,y1-70);
  }
   if (CATEGORIA_QCI == 3)
  {
    text("Valor ref.",xtext12-10,y1-90);
    text("Categoria III",xtext12-25,y1-70);
  }
  smooth();
 }
```

```
void drawEllipses()
{
//Green ellipses
  strokeWeight(2);
  fill(40);
  ellipse(xgreen,y1+10,20,20);
  ellipse(xgreen,y2+10,20,20);
  ellipse(xgreen,y3+10,20,20);
  ellipse(xgreen,y4+10,20,20);
  ellipse(xgreen,y5+10,20,20);

  //Yellow ellipses
  ellipse(xyel,y1+10,20,20);
  ellipse(xyel,y2+10,20,20);
  ellipse(xyel,y3+10,20,20);
  ellipse(xyel,y4+10,20,20);
  ellipse(xyel,y5+10,20,20);

  //Red ellipses
  ellipse(xred,y1+10,20,20);
  ellipse(xred,y2+10,20,20);
  ellipse(xred,y3+10,20,20);
  ellipse(xred,y4+10,20,20);
  ellipse(xred,y5+10,20,20);
}

void IEQ ()
{
  if (CATEGORIA_QCI == 1)//Recommended values for the different parameters
  {
    Tmin = 21;//Celsius
    Tmax = 25;//Celsius
    HRmin = 30;//%
    HRmax = 50;//%
    CO2min = 0;//ppm
    CO2max = 730;//ppm
    Luxmin = 500;//lux
    //Luxmax = 3000;//lux
  }

  if (CATEGORIA_QCI == 2)//Recommended values for the different parameters
  {
    Tmin = 20;//Celsius
    Tmax = 26;//Celsius
    HRmin = 25;//%
    HRmax = 60;//%
    CO2min = 0;//ppm
    CO2max = 880;//ppm
    Luxmin = 500;//lux
  }

  if (CATEGORIA_QCI == 3)//Recommended values for the different parameters
  {
    Tmin = 19;//Celsius
    Tmax = 27;//Celsius
    HRmin = 20;//%
    HRmax = 70;//%
    CO2min = 0;//ppm
    CO2max = 1180;//ppm
    Luxmin = 500;//lux
    }

  if (ESPACO == 'A' || ESPACO == 'B')//Recommended values for the diferente spaces
    {
      Noisemin = 0;//dbA
      Noisemax = 35;//dbA
    }
    if (ESPACO == 'C')
    {
      Noisemin = 0;//dbA
      Noisemax = 45;//dbA
    }

//EVALUATION OF INDOOR ENVIRONMENT QUALITY
  //ADQUATE INDOOR ENVIRONMENT
  if (Temp > Tmin + 1  && Temp < Tmax - 1 )
  {
    fill(0,220,0);
    ellipse(xgreen,y1+10,20,20);
    image(img7, xgreen-15, y1-35);
  }
    if (HR > HRmin + 2 && HR < HRmax - 2)
  {
    fill(0,220,0);
    ellipse(xgreen,y2+10,20,20);
    image(img7, xgreen-15, y2-35);
  }
      if (CO2 >= CO2min && CO2 < CO2max - 100)
  {
    fill(0,220,0);
    ellipse(xgreen,y3+10,20,20);
    image(img7, xgreen-15, y3-35);
  }
    if (Lux >= Luxmin)
  {
    fill(0,220,0);
```

```
    ellipse(xgreen,y4+10,20,20);
    image(img7, xgreen-15, y4-35);
   }
    if (dBA < Noisemax - 2)
   {
    fill(0,220,0);
    ellipse(xgreen,y5+10,20,20);
    image(img7, xgreen-15, y5-35);
   }

   //ACCEPTABLE INDOOR ENVIRONMENT
   if (Temp == Tmin || Temp <= Tmin + 1 && Temp >= Tmin - 1 || Temp == Tmax || Temp <= Tmax + 1 && Temp >=
Tmax - 1)
   {
    fill(250,200,0);
    ellipse(xyel,y1+10,20,20);
    image(img8, xyel-17, y1-35);
   }
   if (HR == HRmin || HR <= HRmin + 2 && HR >= HRmin - 2  || HR == HRmax || HR <= HRmax + 2 && HR >=
HRmax - 2)
   {
    fill(250,200,0);
    ellipse(xyel,y2+10,20,20);
    image(img8, xyel-15, y2-35);
   }
   if (CO2 == CO2max || CO2 >= CO2max -100 && CO2 <= CO2max+100)
   {
    fill(250,200,0);
    ellipse(xyel,y3+10,20,20);
    image(img8, xyel-15, y3-35);
   }
    if (Lux < Luxmin && Lux>=(Luxmin*0.67))
   {
    fill(250,200,0);
    ellipse(xyel,y4+10,20,20);
    image(img8, xyel-15, y4-35);
   }
    if (dBA == Noisemax || dBA >= Noisemax - 2 && dBA <= Noisemax + 2)
   {
    fill(250,200,0);
    ellipse(xyel,y5+10,20,20);
    image(img8, xyel-15, y5-35);
   }
   //INADQUATE INDOOR ENVIRONMENT
    if (Temp < Tmin - 1 || Temp > Tmax + 1 )
   {
    fill(250,0,0);
    ellipse(xred,y1+10,20,20);
    image(img9, xred-17, y1-35);
   }
    if (HR < (HRmin - 2) || HR > (HRmax + 2))
   {
    fill(250,0,0);
    ellipse(xred,y2+10,20,20);
    image(img9, xred-15, y2-35);
   }
    if (CO2 > (CO2max + 100))
   {
    fill(250,0,0);
    ellipse(xred,y3+10,20,20);
    image(img9, xred-15, y3-35);
   }
   if (Lux < Luxmin*0.67)
   {
    fill(250,0,0);
    ellipse(xred,y4+10,20,20);
    image(img9, xred-17, y4-35);
   }
    if (dBA > Noisemax+2)
   {
    fill(250,0,0);
    ellipse(xred,y5+10,20,20);
    image(img9, xred-17, y5-35);
   }
}
}
```

Anexo D.3: Classe para os gráficos inferiores

```
public class plotsDOWN extends NApplet {
int color1=40;//color of the plot shape
int color2=255;// color of labels

//TAB DEFINITIONS
int x1tabA=50;int x1tabB=x1tabA+100;int x1tabC=x1tabB+100;int x1tabD=x1tabC+100;int x1tabE=x1tabD+100;
int y1tab=10;
int x2tabA=140;int x2tabB=x2tabA+100;int x2tabC=x2tabB+100;int x2tabD=x2tabC+100;int x2tabE=x2tabD+100;
int y2tab=40;

//PLOT_A VARIABLES
float dataMinA, dataMaxA;
float plotX1, plotY1A;
float plotX2, plotY2A;
float labelX, labelYA;
```

```
int buttonA=0;
int oldbuttonA=0;
int stateA=0;
int oldstateA=0;
//Set the time data
int timeIntervalA= 1;
int timeMinA = 0;
int timeMaxA = 24;
int timelenghtA = timeMaxA + 1;
int volumeIntervalA = 4;
int volumeIntervalMinorA = 4;
int [] timeA = new int [timelenghtA];
float lineX = 100;//To start next to the Y axis

//PLOT_B VARIABLES
float dataMinB, dataMaxB;
//Set the time data
int volumeIntervalB = 10;
int volumeIntervalMinorB = 10;
int buttonB=0;
int oldbuttonB=0;
int stateB=1;
int oldstateB=0;

//PLOT_C VARIABLES
float dataMinC, dataMaxC;
//Set the time data
int volumeIntervalC = 250;
int volumeIntervalMinorC = 250;
int buttonC=0;
int oldbuttonC=0;
int stateC=0;
int oldstateC=0;

//PLOT_D VARIABLES
float dataMinD, dataMaxD;
//Set the time data
int volumeIntervalD = 100;
int volumeIntervalMinorD = 100;
int buttonD=0;
int oldbuttonD=0;
int stateD=0;
int oldstateD=0;

//PLOT_E VARIABLES
float dataMinE, dataMaxE;
//Set the time data
int volumeIntervalE = 10;
int volumeIntervalMinorE = 10;
int buttonE=0;
int oldbuttonE=0;
int stateE=0;
int oldstateE=0;

//text variables
PFont plotFont;

void setup ()
{
 size(640,360);
 background(255);

 //Plot variables definition
 plotX1 = 80;
 plotX2 = width - 40;
 labelX = 50;

 //set time array
  for (int i = 1; i < timelenghtA; i++)
 {
   timeA[i] = i;
 }

 //PLOT_A (Temperature)
 plotY1A = 80;
 plotY2A = height - 70;
 labelYA = height-30;
 dataMinA = 0;
 dataMaxA = ceil (50/volumeIntervalA)*volumeIntervalA;

 //PLOT_B (Relative Humidity)
 dataMinB = 0;
 dataMaxB = ceil (100/volumeIntervalB)*volumeIntervalB;
 //PLOT_C (CO2 Level)
 dataMinC = 0;
 dataMaxC = ceil (3000/volumeIntervalC)*volumeIntervalC;
 //PLOT_D (Light Intensity)
 dataMinD = 0;
 dataMaxD = ceil (2000/volumeIntervalD)*volumeIntervalD;
 //PLOT_E (Noise Level)
 dataMinE = 0;
 dataMaxE = ceil (90/volumeIntervalE)*volumeIntervalE;

 plotFont = loadFont( "SansSerif.bold-48.vlw" );
 textFont(plotFont);
 smooth();
```

```
}

void draw ()
{
  drawrects();
  drawAround_rects();

  //PLOT_A DRAW
  drawPLOTA ();
  //PLOT_B DRAW
  drawPLOTB();
  //PLOT_C DRAW
  drawPLOTC();
  //PLOT_D DRAW
  drawPLOTD();
  //PLOT_E DRAW
  drawPLOTE();
  //DRAW TABS
  drawTABS();

  //DRAW Middle Line
  stroke(0, 60, 72);
  strokeWeight(10);
  line(0,0,0,height);
  smooth();
}

//TABS
void drawTABS()
{
fill(#4A9999);
rect(x1tabA,y1tab,x2tabA,y2tab);//PLOT_A TAB
rect(x1tabB,y1tab,x2tabB,y2tab);//PLOT_B TAB
rect(x1tabC,y1tab,x2tabC,y2tab);//PLOT_C TAB
rect(x1tabD,y1tab,x2tabD,y2tab);//PLOT_D TAB
rect(x1tabE,y1tab,x2tabE,y2tab);//PLOT_E TAB

//tab's text
textSize(13);
fill(0, 60, 72);
strokeWeight(0);
text("Temperatura",x1tabA+6,y1tab+20);
text("HR",x1tabB+30,y1tab+20);
text("CO2",x1tabC+27,y1tab+20);
text("Iluminância",x1tabD+10,y1tab+20);
text("Ruído",x1tabE+22,y1tab+20);
smooth();
}
```

```
//PLOT GRAPH RECTS
void drawrects()
{
  noFill();
  rectMode(CORNERS);
  stroke(0, 90, 102);
  strokeWeight(1);
  rect(plotX1, plotY1A, plotX2, plotY2A);
  stroke(2);
  smooth();
  fill(#4A8793);
  noStroke();
  rect(0,0,width,plotY1A);
  rect(0,0,plotX1,height);
  rect(0,height,width,plotY2A);
  rect(plotX2,0,width,height);
}

void drawAround_rects()
{
 stroke(2);
  smooth();
  fill(0, 90, 102);
  noStroke();
  rect(0,0,width,plotY1A-0.5);
  rect(0,plotY2A,width,height);
  rect(0,0,plotX1,height);
  rect(plotX2,0,width,height);
}

//PLOT_A FUNCTIONS
void drawTitleA() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Temperatura";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsA()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
```

```
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
   }
}

  void drawAxisLabelsA()
{
  fill(color2);
  textSize(13);
  textLeading(15);


  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Graus Celsius (ºC)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsA() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinA; v <= dataMaxA; v += volumeIntervalMinorA) {
    if (v % volumeIntervalMinorA == 0) {// If a tick mark
      float y = map(v, dataMinA, dataMaxA, plotY2A, plotY1A);
      if (v % volumeIntervalA == 0) {// If a major tick mark
        float textOffset = textAscent()/2;//Center vertically
        if (v == dataMinA) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxA) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);// Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
      }
    }
}

void drawDataLineA() {
  stroke(color1);
  strokeWeight(1);
  float Tempmap = map (Temp,0,48,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,Tempmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//PLOT_B FUNCTIONS
void drawTitleB() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Humidade Relativa";
  text(title, plotX1, plotY1A - 10);
  smooth();
}

void drawYearLabelsB()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
```

```
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsB()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Humidade Relativa (%)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsB() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinB; v <= dataMaxB; v += volumeIntervalMinorB) {
    if (v % volumeIntervalMinorB == 0) {// If a tick mark
      float y = map(v, dataMinB, dataMaxB, plotY2A, plotY1A);
      if (v % volumeIntervalB == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// Center vertically
        if (v == dataMinB) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxB) {
          textOffset = textAscent()-5;//Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);// Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
      }
    }
  }
}

void drawDataLineB() {
  stroke(color1);
  strokeWeight(1);
  float HRmap = map (HR,0,100,plotY2A,plotY1A); //map the respective value received from serial port according to y
axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,HRmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
  }
}

//PLOT_C FUNCTIONS

void drawTitleC() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Nível de CO2";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsC()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
```

```
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsC()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX - 20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Concentração de CO2(ppm)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsC() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinC; v <= dataMaxC; v += volumeIntervalMinorC) {
    if (v % volumeIntervalMinorC == 0) {// If a tick mark
      float y = map(v, dataMinC, dataMaxC, plotY2A, plotY1A);
      if (v % volumeIntervalC == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// Center vertically
        if (v == dataMinC) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxC) {
          textOffset = textAscent()-5;//Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);     // Draw major tick
        line(plotX1-2, y+8, plotX1, y+8);  //line to draw midle lines
      }
    }
  }
}
```

```
void drawDataLineC() {
  stroke(color1);
  strokeWeight(1);
  float CO2map = map (CO2,0,3000,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,CO2map); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
  }
}

//PLOT_D FUNCTIONS
void drawTitleD() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Iluminância";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsD()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}
```

```
  void drawAxisLabelsD()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX - 20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Iluminância (lux)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsD() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinD; v <= dataMaxD; v += volumeIntervalMinorD) {
    if (v % volumeIntervalMinorD == 0) {// If a tick mark
      float y = map(v, dataMinD, dataMaxD, plotY2A, plotY1A);
      if (v % volumeIntervalD == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// CENTER vertically
        if (v == dataMinD) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxD) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);     // Draw major tick
        line(plotX1-2, y+5, plotX1, y+5);  //line to draw midle lines
      }
    }
  }
}

void drawDataLineD() {
  stroke(color1);
  strokeWeight(1);
  float Luxmap = map (Lux,0,2000,plotY2A,plotY1A);//map the respective value received from serial port according to
y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,Luxmap); //draw the graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//PLOT_E FUNCTIONS
void drawTitleE() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Nível de ruído";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsE()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsE()
{
  fill(color2);
```

```
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Nível de ruído (dBA)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsE() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinE; v <= dataMaxE; v += volumeIntervalMinorE) {
    if (v % volumeIntervalMinorE == 0) { // If a tick mark
      float y = map(v, dataMinE, dataMaxE, plotY2A, plotY1A);
      if (v % volumeIntervalE == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// CENTER vertically
        if (v == dataMinE) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxE) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y); //Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
      }
    }
  }
}

void drawDataLineE() {
  stroke(color1);
  strokeWeight(1);
  float NoiseLmap = map (dBA,0,90,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,NoiseLmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//CHOOSE PLOT FUNCTIONS
//FUNTION TO CLEAN BACKGROUND
void cleanBackground()
{
   loadPixels();
   for (int i = 0; i < width * height ; i++) pixels[i] = 0;
   updatePixels();
}

void mousePressed()
{
  println("Coordinates: " + mouseX +"," + mouseY);
  mouseActionA();
  mouseActionB();
  mouseActionC();
  mouseActionD();
  mouseActionE();
}

void mouseActionA()
{
  if( mouseX > x1tabA && mouseX < x2tabA)
  {
    if( mouseY > y1tab && mouseY < y2tab)
    {
      buttonA=1;
      oldbuttonA=0;
      cleanBackground();
      background(255);
    }
  }
}

void mouseActionB()
{
  if( mouseX > x1tabB && mouseX < x2tabB)
  {
    if( mouseY > y1tab && mouseY < y2tab)
    {
      buttonB=1;
```

```
    oldbuttonB=0;
    cleanBackground();
    background(255);
   }
  }
}

void mouseActionC()
{
 if( mouseX > x1tabC && mouseX < x2tabC)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonC=1;
   oldbuttonC=0;
   cleanBackground();
   background(255);
  }
 }
}

void mouseActionD()
{
 if( mouseX > x1tabD && mouseX < x2tabD)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonD=1;
   oldbuttonD=0;
   cleanBackground();
   background(255);
  }
 }
}

void mouseActionE()
{
 if( mouseX > x1tabE && mouseX < x2tabE)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonE=1;
   oldbuttonE=0;
   cleanBackground();
   background(255);
  }
 }
}
```

```
void drawPLOTA ()
{
 if(buttonA==1 && oldbuttonA==0)
 {
  stateA=1-stateA;
 }
  oldbuttonA=buttonA;

 if (stateA==1)
 {
  drawrects();
  drawAround_rects();
  drawAxisLabelsA();
  drawYearLabelsA();
  drawVolumeLabelsA();
  drawTitleA();
  drawDataLineA();
  noFill();
  noStroke();
  rect((width/2)-25,20,width,plotY2A);
  rect(plotX1,plotY1A,plotX2,plotY2A);
 }
}

void drawPLOTB ()
{
 if(buttonB==1 && oldbuttonB==0)
 {
  stateB=1-stateB;
 }
  oldbuttonB=buttonB;

 if (stateB==1)
 {
  drawrects();
  drawAround_rects();
  drawAxisLabelsB();
  drawYearLabelsB();
  drawVolumeLabelsB();
  drawTitleB();
  drawDataLineB();
  noFill();
  noStroke();
  rect((width/2)-25,20,width,plotY2A);
  rect(plotX1,plotY1A,plotX2,plotY2A);
 }
}
```

```
void drawPLOTC ()
{
  if(buttonC==1 && oldbuttonC==0)
  {
    stateC=1-stateC;
  }
    oldbuttonC=buttonC;

  if (stateC==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsC();
   drawYearLabelsC();
   drawVolumeLabelsC();
   drawTitleC();
   drawDataLineC();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}

void drawPLOTD ()
{
  if(buttonD==1 && oldbuttonD==0)
  {
    stateD=1-stateD;
  }
    oldbuttonD=buttonD;

  if (stateD==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsD();
   drawYearLabelsD();
   drawVolumeLabelsD();
   drawTitleD();
   drawDataLineD();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}
```

```
void drawPLOTE ()
{
  if(buttonE==1 && oldbuttonE==0)
  {
    stateE=1-stateE;
  }
    oldbuttonE=buttonE;

  if (stateE==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsE();
   drawYearLabelsE();
   drawVolumeLabelsE();
   drawTitleE();
   drawDataLineE();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}
}
```

Anexo D.4: Classe para os gráficos superiores

```
public class plotsDOWN extends NApplet {
int color1=40;//color of the plot shape
int color2=255;// color of labels

//TAB DEFINITIONS
int x1tabA=50;int x1tabB=x1tabA+100;int x1tabC=x1tabB+100;int x1tabD=x1tabC+100;int x1tabE=x1tabD+100;
int y1tab=10;
int x2tabA=140;int x2tabB=x2tabA+100;int x2tabC=x2tabB+100;int x2tabD=x2tabC+100;int x2tabE=x2tabD+100;
int y2tab=40;

//PLOT_A VARIABLES
float dataMinA, dataMaxA;
```

```
float plotX1, plotY1A;
float plotX2, plotY2A;
float labelX, labelYA;
int buttonA=0;
int oldbuttonA=0;
int stateA=0;
int oldstateA=0;
//Set the time data
int timeIntervalA= 1;
int timeMinA = 0;
int timeMaxA = 24;
int timelenghtA = timeMaxA + 1;
int volumeIntervalA = 4;
int volumeIntervalMinorA = 4;
int [] timeA = new int [timelenghtA];
float lineX = 100;//To start next to the Y axis

//PLOT_B VARIABLES
float dataMinB, dataMaxB;
//Set the time data
int volumeIntervalB = 10;
int volumeIntervalMinorB = 10;
int buttonB=0;
int oldbuttonB=0;
int stateB=1;
int oldstateB=0;

//PLOT_C VARIABLES
float dataMinC, dataMaxC;
//Set the time data
int volumeIntervalC = 250;
int volumeIntervalMinorC = 250;
int buttonC=0;
int oldbuttonC=0;
int stateC=0;
int oldstateC=0;

//PLOT_D VARIABLES
float dataMinD, dataMaxD;
//Set the time data
int volumeIntervalD = 100;
int volumeIntervalMinorD = 100;
int buttonD=0;
int oldbuttonD=0;
int stateD=0;
int oldstateD=0;

//PLOT_E VARIABLES
float dataMinE, dataMaxE;
//Set the time data
int volumeIntervalE = 10;
int volumeIntervalMinorE = 10;
int buttonE=0;
int oldbuttonE=0;
int stateE=0;
int oldstateE=0;

//text variables
PFont plotFont;

void setup ()
{
 size(640,360);
 background(255);

 //Plot variables definition
 plotX1 = 80;
 plotX2 = width - 40;
 labelX = 50;

 //set time array
  for (int i = 1; i < timelenghtA; i++)
 {
   timeA[i] = i;
 }

 //PLOT_A (Temperature)
 plotY1A = 80;
 plotY2A = height - 70;
 labelYA = height-30;
 dataMinA = 0;
 dataMaxA = ceil (50/volumeIntervalA)*volumeIntervalA;

 //PLOT_B (Relative Humidity)
 dataMinB = 0;
 dataMaxB = ceil (100/volumeIntervalB)*volumeIntervalB;
 //PLOT_C (CO2 Level)
 dataMinC = 0;
 dataMaxC = ceil (3000/volumeIntervalC)*volumeIntervalC;
 //PLOT_D (Light Intensity)
 dataMinD = 0;
 dataMaxD = ceil (2000/volumeIntervalD)*volumeIntervalD;
 //PLOT_E (Noise Level)
 dataMinE = 0;
 dataMaxE = ceil (90/volumeIntervalE)*volumeIntervalE;
```

```
  plotFont = loadFont( "SansSerif.bold-48.vlw" );
  textFont(plotFont);
  smooth();
}

void draw ()
{
  drawrects();
  drawAround_rects();

  //PLOT_A DRAW
  drawPLOTA ();
  //PLOT_B DRAW
  drawPLOTB();
  //PLOT_C DRAW
  drawPLOTC();
  //PLOT_D DRAW
  drawPLOTD();
  //PLOT_E DRAW
  drawPLOTE();
  //DRAW TABS
  drawTABS();

  //DRAW Middle Line
  stroke(0, 60, 72);
  strokeWeight(10);
  line(0,0,0,height);
  smooth();
}

//TABS
void drawTABS()
{
fill(#4A9999);
rect(x1tabA,y1tab,x2tabA,y2tab);//PLOT_A TAB
rect(x1tabB,y1tab,x2tabB,y2tab);//PLOT_B TAB
rect(x1tabC,y1tab,x2tabC,y2tab);//PLOT_C TAB
rect(x1tabD,y1tab,x2tabD,y2tab);//PLOT_D TAB
rect(x1tabE,y1tab,x2tabE,y2tab);//PLOT_E TAB

//tab's text
textSize(13);
fill(0, 60, 72);
strokeWeight(0);
text("Temperatura",x1tabA+6,y1tab+20);
text("HR",x1tabB+30,y1tab+20);
text("CO2",x1tabC+27,y1tab+20);
text("Iluminância",x1tabD+10,y1tab+20);
text("Ruído",x1tabE+22,y1tab+20);
smooth();
}

//PLOT GRAPH RECTS
void drawrects()
{
  noFill();
  rectMode(CORNERS);
  stroke(0, 90, 102);
  strokeWeight(1);
  rect(plotX1, plotY1A, plotX2, plotY2A);
  stroke(2);
  smooth();
  fill(#4A8793);
  noStroke();
  rect(0,0,width,plotY1A);
  rect(0,0,plotX1,height);
  rect(0,height,width,plotY2A);
  rect(plotX2,0,width,height);
}

void drawAround_rects()
{
 stroke(2);
  smooth();
  fill(0, 90, 102);
  noStroke();
  rect(0,0,width,plotY1A-0.5);
  rect(0,plotY2A,width,height);
  rect(0,0,plotX1,height);
  rect(plotX2,0,width,height);
}

//PLOT_A FUNCTIONS
void drawTitleA() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Temperatura";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsA()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);
```

```
  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsA()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Graus Celsius (ºC)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsA() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinA; v <= dataMaxA; v += volumeIntervalMinorA) {
    if (v % volumeIntervalMinorA == 0) {// If a tick mark
      float y = map(v, dataMinA, dataMaxA, plotY2A, plotY1A);
      if (v % volumeIntervalA == 0) {// If a major tick mark
        float textOffset = textAscent()/2;//Center vertically
        if (v == dataMinA) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxA) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);// Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
    }
  }
}

void drawDataLineA() {
  stroke(color1);
  strokeWeight(1);
  float Tempmap = map (Temp,0,48,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,Tempmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
  }
}

//PLOT_B FUNCTIONS
void drawTitleB() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Humidade Relativa";
  text(title, plotX1, plotY1A - 10);
  smooth();
}

void drawYearLabelsB()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
```

```
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsB()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Humidade Relativa (%)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsB() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinB; v <= dataMaxB; v += volumeIntervalMinorB) {
    if (v % volumeIntervalMinorB == 0) {// If a tick mark
      float y = map(v, dataMinB, dataMaxB, plotY2A, plotY1A);
      if (v % volumeIntervalB == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// Center vertically
        if (v == dataMinB) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxB) {
          textOffset = textAscent()-5;//Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);// Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
      }
    }
  }
}

void drawDataLineB() {
  stroke(color1);
  strokeWeight(1);
  float HRmap = map (HR,0,100,plotY2A,plotY1A); //map the respective value received from serial port according to y
axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,HRmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
  }
}

//PLOT_C FUNCTIONS
void drawTitleC() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Nível de CO2";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsC()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
```

```
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsC()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX - 20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Concentração de CO2(ppm)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsC() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinC; v <= dataMaxC; v += volumeIntervalMinorC) {
    if (v % volumeIntervalMinorC == 0) {// If a tick mark
      float y = map(v, dataMinC, dataMaxC, plotY2A, plotY1A);
      if (v % volumeIntervalC == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// Center vertically
        if (v == dataMinC) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxC) {
          textOffset = textAscent()-5;//Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);     // Draw major tick
        line(plotX1-2, y+8, plotX1, y+8);  //line to draw midle lines
      }
    }
  }
}

void drawDataLineC() {
  stroke(color1);
  strokeWeight(1);
  float CO2map = map (CO2,0,3000,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,CO2map); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//PLOT_D FUNCTIONS
void drawTitleD() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Iluminância";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsD()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
```

```
}

  void drawAxisLabelsD()
{
  fill(color2);
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX - 20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Iluminância (lux)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsD() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinD; v <= dataMaxD; v += volumeIntervalMinorD) {
    if (v % volumeIntervalMinorD == 0) {// If a tick mark
      float y = map(v, dataMinD, dataMaxD, plotY2A, plotY1A);
      if (v % volumeIntervalD == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// CENTER vertically
        if (v == dataMinD) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxD) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y);     // Draw major tick
        line(plotX1-2, y+5, plotX1, y+5);  //line to draw midle lines
      }
    }
  }
}

void drawDataLineD() {
  stroke(color1);
  strokeWeight(1);
```

```
  float Luxmap = map (Lux,0,2000,plotY2A,plotY1A);//map the respective value received from serial port according to
y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,Luxmap); //draw the graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//PLOT_E FUNCTIONS
void drawTitleE() {
  fill(color2);
  textSize(20);
  textAlign(LEFT);
  String title = "Nível de ruído";
  text(title, plotX1, plotY1A - 10);
}

void drawYearLabelsE()
{
  fill(color2);
  textSize(10);
  textAlign(CENTER);

  // Use thin, blue lines to draw the grid
  for (int row = 0; row < timelenghtA; row++) {
    if (timeA[row] % timeIntervalA == 0) {
      float x = map(timeA[row], timeMinA, timeMaxA, plotX1, plotX2);
      text(timeA[row], x, plotY2A + textAscent() + 10);
      stroke(color1);
      strokeWeight(0.00001);
      line(x, plotY1A, x, plotY2A);
      line(plotX1,plotY2A,plotX2,plotY2A);
      line(plotX2,plotY1A,plotX2,plotY2A);
      stroke(0);
      strokeWeight(1);
      line(x, plotY2A+4, x, plotY2A);
    }
  }
}

  void drawAxisLabelsE()
{
  fill(color2);
```

```
  textSize(13);
  textLeading(15);

  //Rotate text 90º necessary functions marked with 1
  pushMatrix();//1
  translate(labelX-20, (plotY1A+plotY2A)/2);//1
  rotate(-HALF_PI);//1
  textAlign(CENTER, CENTER);
  text("Nível de ruído (dBA)", 0,0);
  textAlign(CENTER);
  popMatrix();//1
  textAlign(CENTER);
  text("Tempo (h)", (plotX1+plotX2)/2, labelYA);
}

void drawVolumeLabelsE() {
  fill(color2);
  textSize(10);
  textAlign(RIGHT);
  stroke(0);
  strokeWeight(1);

  for (float v = dataMinE; v <= dataMaxE; v += volumeIntervalMinorE) {
    if (v % volumeIntervalMinorE == 0) { // If a tick mark
      float y = map(v, dataMinE, dataMaxE, plotY2A, plotY1A);
      if (v % volumeIntervalE == 0) {// If a major tick mark
        float textOffset = textAscent()/2;// CENTER vertically
        if (v == dataMinE) {
          textOffset = textAscent()-2;// Align by the bottom
        } else if (v == dataMaxE) {
          textOffset = textAscent()-5;// Align by the top
        }
        text(floor(v), plotX1 - 10, y + textOffset);
        line(plotX1 - 4, y, plotX1, y); //Draw major tick
        line(plotX1-2, y+9, plotX1, y+9);//line to draw midle lines
      }
    }
  }
}

void drawDataLineE() {
  stroke(color1);
  strokeWeight(1);
  float NoiseLmap = map (dBA,0,90,plotY2A,plotY1A);//map the respective value received from serial port according
to y axis
  float maptime=map(timeX,0,86400,plotX1,plotX2); //map the respective value received from serial port according to
x axis
  line (maptime,plotY2A,maptime,NoiseLmap); //draw graph shape
  smooth();

  if (timeX==86340)//One minute before midnight erase the graph
  {
    background (255);
   }
}

//CHOOSE PLOT FUNCTIONS
//FUNTION TO CLEAN BACKGROUND
void cleanBackground()
{
   loadPixels();
   for (int i = 0; i < width * height ; i++) pixels[i] = 0;
   updatePixels();
}

void mousePressed()
{
  println("Coordinates: " + mouseX +"," + mouseY);
  mouseActionA();
  mouseActionB();
  mouseActionC();
  mouseActionD();
  mouseActionE();
}

void mouseActionA()
{
  if( mouseX > x1tabA && mouseX < x2tabA)
  {
    if( mouseY > y1tab && mouseY < y2tab)
    {
      buttonA=1;
      oldbuttonA=0;
      cleanBackground();
      background(255);
    }
  }
}

void mouseActionB()
{
  if( mouseX > x1tabB && mouseX < x2tabB)
  {
    if( mouseY > y1tab && mouseY < y2tab)
    {
      buttonB=1;
```

```
    oldbuttonB=0;
    cleanBackground();
    background(255);
   }
  }
}

void mouseActionC()
{
 if( mouseX > x1tabC && mouseX < x2tabC)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonC=1;
   oldbuttonC=0;
   cleanBackground();
   background(255);
  }
 }
}

void mouseActionD()
{
 if( mouseX > x1tabD && mouseX < x2tabD)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonD=1;
   oldbuttonD=0;
   cleanBackground();
   background(255);
  }
 }
}

void mouseActionE()
{
 if( mouseX > x1tabE && mouseX < x2tabE)
 {
  if( mouseY > y1tab && mouseY < y2tab)
  {
   buttonE=1;
   oldbuttonE=0;
   cleanBackground();
   background(255);
  }
 }
}
```

```
void drawPLOTA ()
{
 if(buttonA==1 && oldbuttonA==0)
 {
  stateA=1-stateA;
 }
  oldbuttonA=buttonA;

 if (stateA==1)
 {
  drawrects();
  drawAround_rects();
  drawAxisLabelsA();
  drawYearLabelsA();
  drawVolumeLabelsA();
  drawTitleA();
  drawDataLineA();
  noFill();
  noStroke();
  rect((width/2)-25,20,width,plotY2A);
  rect(plotX1,plotY1A,plotX2,plotY2A);
 }
}

void drawPLOTB ()
{
 if(buttonB==1 && oldbuttonB==0)
 {
  stateB=1-stateB;
 }
  oldbuttonB=buttonB;

 if (stateB==1)
 {
  drawrects();
  drawAround_rects();
  drawAxisLabelsB();
  drawYearLabelsB();
  drawVolumeLabelsB();
  drawTitleB();
  drawDataLineB();
  noFill();
  noStroke();
  rect((width/2)-25,20,width,plotY2A);
  rect(plotX1,plotY1A,plotX2,plotY2A);
 }
}
```

```
void drawPLOTC ()
{
  if(buttonC==1 && oldbuttonC==0)
  {
    stateC=1-stateC;
  }
    oldbuttonC=buttonC;

  if (stateC==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsC();
   drawYearLabelsC();
   drawVolumeLabelsC();
   drawTitleC();
   drawDataLineC();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}

void drawPLOTD ()
{
  if(buttonD==1 && oldbuttonD==0)
  {
    stateD=1-stateD;
  }
    oldbuttonD=buttonD;

  if (stateD==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsD();
   drawYearLabelsD();
   drawVolumeLabelsD();
   drawTitleD();
   drawDataLineD();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}
```

```
void drawPLOTE ()
{
  if(buttonE==1 && oldbuttonE==0)
  {
    stateE=1-stateE;
  }
    oldbuttonE=buttonE;

  if (stateE==1)
  {
   drawrects();
   drawAround_rects();
   drawAxisLabelsE();
   drawYearLabelsE();
   drawVolumeLabelsE();
   drawTitleE();
   drawDataLineE();
   noFill();
   noStroke();
   rect((width/2)-25,20,width,plotY2A);
   rect(plotX1,plotY1A,plotX2,plotY2A);
  }
}
}
```